Introducción - Modelo Relacional - SQL

Juan F. Pérez

Departamento MACC
Matemáticas Aplicadas y Ciencias de la Computación
Universidad del Rosario

juanferna.perez@urosario.edu.co

Primer Semestre de 2019

Contenidos

1. Introducción
2. Bases de datos
   - Abstracción y modelos de datos
   - Lenguajes de una base de datos
   - Diseño
3. Modelo relacional
4. Álgebra relacional
5. Álgebra relacional vs. SQL
6. Resumen

Introducción

Introducción

- Bienvenidos al curso «Manejo de bases de datos»

Introducción

- Bienvenidos al curso «Manejo de bases de datos»
- ¿De qué se trata este curso?

Introducción

- Bienvenidos al curso «Manejo de bases de datos»
- ¿De qué se trata este curso?
- ¿De qué no se trata este curso?

Introducción

- Bienvenidos al curso «Manejo de bases de datos»
- ¿De qué se trata este curso?
- ¿De qué no se trata este curso?
- Dos sesiones por semana:
  - Mi 1pm - 3pm @ AdaL: 2 horas teóricas
  - Vi 1pm - 4pm @ AdaL: 3 horas taller (trabajo individual, en grupo, tareas, proyectos)

Introducción

- Bienvenidos al curso «Manejo de bases de datos»
- ¿De qué se trata este curso?
- ¿De qué no se trata este curso?
- Dos sesiones por semana:
  - Mi 1pm - 3pm @ AdaL: 2 horas teóricas
  - Vi 1pm - 4pm @ AdaL: 3 horas taller (trabajo individual, en grupo, tareas, proyectos)
- Horario de atención: Ju 4:30pm - 6:30pm @ Cabal-504

Introducción

- Bienvenidos al curso «Manejo de bases de datos»
- ¿De qué se trata este curso?
- ¿De qué no se trata este curso?
- Dos sesiones por semana:
  - Mi 1pm - 3pm @ AdaL: 2 horas teóricas
  - Vi 1pm - 4pm @ AdaL: 3 horas taller (trabajo individual, en grupo, tareas, proyectos)
- Horario de atención: Ju 4:30pm - 6:30pm @ Cabal-504
- Guía de asignatura en e-aulas

Plan de vuelo

5 módulos:

- M1: Bases de Datos Relacionales (Parcial 1)

Plan de vuelo

5 módulos:

- M1: Bases de Datos Relacionales (Parcial 1)
- M2: Diseño de Bases de Datos (Parcial 2)

Plan de vuelo

5 módulos:

- M1: Bases de Datos Relacionales (Parcial 1)
- M2: Diseño de Bases de Datos (Parcial 2)
- M3: Almacenamiento de datos y consultas

Plan de vuelo

5 módulos:

- M1: Bases de Datos Relacionales (Parcial 1)
- M2: Diseño de Bases de Datos (Parcial 2)
- M3: Almacenamiento de datos y consultas
- M4: Administración de transacciones

Plan de vuelo

5 módulos:

- M1: Bases de Datos Relacionales (Parcial 1)
- M2: Diseño de Bases de Datos (Parcial 2)
- M3: Almacenamiento de datos y consultas
- M4: Administración de transacciones
- M5: Arquitectura de sistemas (Parcial 3 - Fecha examen final)

Calificación

- 3 parciales (20 % c/u): 60 %

Calificación

- 3 parciales (20 % c/u): 60 %
- Tareas (5 - 8): 15 %

Calificación

- 3 parciales (20 % c/u): 60 %
- Tareas (5 - 8): 15 %
- Proyectos (3 - 5): 25 %

Bases de datos

Bases de datos

- Colecciones organizadas de datos inter-relacionados que representan (modelan) algún aspecto del mundo real

Bases de datos

- Colecciones organizadas de datos inter-relacionados que representan (modelan) algún aspecto del mundo real
- ¿Ejemplos?

Bases de datos

- Colecciones organizadas de datos inter-relacionados que representan (modelan) algún aspecto del mundo real
- ¿Ejemplos?
- DBMS: database management system: software

Bases de datos

- Colecciones organizadas de datos inter-relacionados que representan (modelan) algún aspecto del mundo real
- ¿Ejemplos?
- DBMS: database management system: software
- Parte fundamental de (casi) cualquier aplicación de software

Ejemplo

- Estudiantes de la universidad

Ejemplo

- Estudiantes de la universidad
- ¿Cómo almacenamos esta información?

Ejemplo

- Estudiantes de la universidad
- ¿Cómo almacenamos esta información?
- Archivos de texto (planos): comma-separated value (CSV)

Ejemplo

- Estudiantes de la universidad
- ¿Cómo almacenamos esta información?
- Archivos de texto (planos): comma-separated value (CSV)
- Archivo con datos de estudiantes, otro archivo con datos de cursos dictados

Ejemplo

- Estudiantes de la universidad
- ¿Cómo almacenamos esta información?
- Archivos de texto (planos): comma-separated value (CSV)
- Archivo con datos de estudiantes, otro archivo con datos de cursos dictados

estudiantes.txt:

```
"Nombre", "Apellidos", "ID"
"Juan", "Pérez", "8888"
"Ana", "Gutiérrez", "5555"
```

cursos.txt

```
"Código", "Nombre", "Fecha"
"MACC01", "Programación", "
    2017-2S"
"MACC02", "Arquitectura", "
    2018-2S"
```

Ejemplo

- ¿Cómo procesar estos archivos?

Ejemplo

- ¿Cómo procesar estos archivos?
- ¿Cómo resolver una consulta? E.g., obtener nombre y apellido de una estudiante dado su número de cédula

Ejemplo

- ¿Cómo procesar estos archivos?
- ¿Cómo resolver una consulta? E.g., obtener nombre y apellido de una estudiante dado su número de cédula
- ¿Eficiencia? ¿Búsqueda de un registro?

Ejemplo

- ¿Cómo procesar estos archivos?
- ¿Cómo resolver una consulta? E.g., obtener nombre y apellido de una estudiante dado su número de cédula
- ¿Eficiencia? ¿Búsqueda de un registro?
- ¿Cómo asociar la lista de estudiantes que toman un curso?

Ejemplo

- ¿Cómo procesar estos archivos?
- ¿Cómo resolver una consulta? E.g., obtener nombre y apellido de una estudiante dado su número de cédula
- ¿Eficiencia? ¿Búsqueda de un registro?
- ¿Cómo asociar la lista de estudiantes que toman un curso?
- Consistencia: garantizar que los datos siempre satisfagan algunas restricciones (e.g., valores no negativos)

Otras preguntas

- ¿Cómo asegurar que un estudiante siempre sea referido con el nombre adecuado? ¿typos?

Otras preguntas

- ¿Cómo asegurar que un estudiante siempre sea referido con el nombre adecuado? ¿typos?
- ¿Cómo representar que un curso puede ser dictado por más de un profesor?

Otras preguntas

- ¿Cómo asegurar que un estudiante siempre sea referido con el nombre adecuado? ¿typos?
- ¿Cómo representar que un curso puede ser dictado por más de un profesor?
- ¿Cómo evitar que alguien abra un archivo y cambie la información por cualquier cosa?

Otras preguntas

- ¿Cómo asegurar que un estudiante siempre sea referido con el nombre adecuado? ¿typos?
- ¿Cómo representar que un curso puede ser dictado por más de un profesor?
- ¿Cómo evitar que alguien abra un archivo y cambie la información por cualquier cosa?
- ¿Cómo permitir que más de una aplicación pueda acceder a estos datos? ¿Replicar código? ¿Coordinación si se (sobre)-escribe información (concurrencia)?

Otras preguntas

- ¿Cómo asegurar que un estudiante siempre sea referido con el nombre adecuado? ¿typos?
- ¿Cómo representar que un curso puede ser dictado por más de un profesor?
- ¿Cómo evitar que alguien abra un archivo y cambie la información por cualquier cosa?
- ¿Cómo permitir que más de una aplicación pueda acceder a estos datos? ¿Replicar código? ¿Coordinación si se (sobre)-escribe información (concurrencia)?
- ¿Cómo permitir que nuevas aplicaciones puedan acceder a los datos sin haber diseñado la base de datos para cada posible aplicación?

Más preguntas

- ¿Cómo garantizar que los datos no se pierdan si el equipo deja de funcionar temporalmente?

Más preguntas

- ¿Cómo garantizar que los datos no se pierdan si el equipo deja de funcionar temporalmente?
- ¿Cómo replicar información la información en varios equipos para garantizar que no se pierda la información si uno o varios equipos fallan definitivamente?

Más preguntas

- ¿Cómo garantizar que los datos no se pierdan si el equipo deja de funcionar temporalmente?
- ¿Cómo replicar información la información en varios equipos para garantizar que no se pierda la información si uno o varios equipos fallan definitivamente?
- Durabilidad: lograr que los datos sigan siendo accesibles, sin errores y sin cambios no solicitados

Más preguntas

- ¿Cómo garantizar que los datos no se pierdan si el equipo deja de funcionar temporalmente?
- ¿Cómo replicar información la información en varios equipos para garantizar que no se pierda la información si uno o varios equipos fallan definitivamente?
- Durabilidad: lograr que los datos sigan siendo accesibles, sin errores y sin cambios no solicitados
- Atomicidad: lograr que un cambio en la dase de datos se ejecute completamente y no solo una parte (e.g., crédito y débito)

Más preguntas

- ¿Cómo garantizar que los datos no se pierdan si el equipo deja de funcionar temporalmente?
- ¿Cómo replicar información la información en varios equipos para garantizar que no se pierda la información si uno o varios equipos fallan definitivamente?
- Durabilidad: lograr que los datos sigan siendo accesibles, sin errores y sin cambios no solicitados
- Atomicidad: lograr que un cambio en la dase de datos se ejecute completamente y no solo una parte (e.g., crédito y débito)
- Respuesta: DBMS (sistema de administración de bases de datos)

DMBS

- Sistema de administración de bases de datos

DMBS

- Sistema de administración de bases de datos
- Permite almacenar y extraer información en una base de datos

DMBS

- Sistema de administración de bases de datos
- Permite almacenar y extraer información en una base de datos
- Permite definir, crear, consultar, actualizar y administrar bases de datos

DMBS

- Sistema de administración de bases de datos
- Permite almacenar y extraer información en una base de datos
- Permite definir, crear, consultar, actualizar y administrar bases de datos
- Se encarga de garantizar durabilidad de la información, mientras otras aplicaciones se encargan de su propia lógica (de negocio)

Algo de historia

- 60s-70s: bases de datos iniciales

Algo de historia

- 60s-70s: bases de datos iniciales
- Difíciles de crear y mantener - capas física (representación física de la información) y lógica unidas

Algo de historia

- 60s-70s: bases de datos iniciales
- Difíciles de crear y mantener - capas física (representación física de la información) y lógica unidas
- Representación física determinaba las consultas posibles

Algo de historia

- 60s-70s: bases de datos iniciales
- Difíciles de crear y mantener - capas física (representación física de la información) y lógica unidas
- Representación física determinaba las consultas posibles
- Necesario saber qué consultas se requerían antes de diseñar la base de datos

Algo de historia

- 60s-70s: bases de datos iniciales
- Difíciles de crear y mantener - capas física (representación física de la información) y lógica unidas
- Representación física determinaba las consultas posibles
- Necesario saber qué consultas se requerían antes de diseñar la base de datos
- Muy difícil extenderlas después de desplegadas

Algo de historia

- Edgar F. Codd (IBM Research Lab)

Algo de historia

- Edgar F. Codd (IBM Research Lab)
- E.F.Codd, «Derivability, redundancy and consistency of relations stored in large data banks», 1969.

Algo de historia

- Edgar F. Codd (IBM Research Lab)
- E.F.Codd, «Derivability, redundancy and consistency of relations stored in large data banks», 1969.
- E.F.Codd, «A relational model of data for large data banks», Information Retrieval, 1970.

Algo de historia

- Edgar F. Codd (IBM Research Lab)
- E.F.Codd, «Derivability, redundancy and consistency of relations stored in large data banks», 1969.
- E.F.Codd, «A relational model of data for large data banks», Information Retrieval, 1970.
- Modelo relacional

Algo de historia

Modelo relacional (abstracción):

- Almacenar la base de datos en estructuras de datos simples (relaciones entre entidades)

Algo de historia

Modelo relacional (abstracción):

- Almacenar la base de datos en estructuras de datos simples (relaciones entre entidades)
- Acceder a los datos a través de un lenguaje de alto nivel (sin referencia a cómo están almacenados los datos)

Algo de historia

Modelo relacional (abstracción):

- Almacenar la base de datos en estructuras de datos simples (relaciones entre entidades)
- Acceder a los datos a través de un lenguaje de alto nivel (sin referencia a cómo están almacenados los datos)
- Almacenamiento físico separado de la lógica de acceso de datos (problema que se resuelve en la implementación)

Bases de datos
Abstracción y modelos de datos

Abstracción de los datos

- **Nivel físico**: cómo se almacenan físicamente los datos

Abstracción de los datos

- **Nivel físico**: cómo se almacenan físicamente los datos
- **Nivel lógico**: qué datos se almacenan y cómo están relacionados. Describe toda la base de datos. Independiente del nivel físico.

Abstracción de los datos

- **Nivel físico**: cómo se almacenan físicamente los datos
- **Nivel lógico**: qué datos se almacenan y cómo están relacionados. Describe toda la base de datos. Independiente del nivel físico.
- **Nivel de vista**: describe solo una parte de la base de datos relevante para una clase de usuarios. Se puede contar con muchas vistas de una misma base de datos.

Abstracción de los datos

- **Nivel físico**: cómo se almacenan físicamente los datos
- **Nivel lógico**: qué datos se almacenan y cómo están relacionados. Describe toda la base de datos. Independiente del nivel físico.
- **Nivel de vista**: describe solo una parte de la base de datos relevante para una clase de usuarios. Se puede contar con muchas vistas de una misma base de datos.
- Aplicaciones trabajan sobre el esquema lógico: independencia del nivel físico

Instancias y esquemas

- **Instancia**: la información almacenada en una base de datos en un momento dado. Cambia continuamente.

Instancias y esquemas

- **Instancia**: la información almacenada en una base de datos en un momento dado. Cambia continuamente.
- **Esquema**: diseño completo de la base de datos. Cambia poco o nada.

Instancias y esquemas

- **Instancia**: la información almacenada en una base de datos en un momento dado. Cambia continuamente.
- **Esquema**: diseño completo de la base de datos. Cambia poco o nada.
- **Esquema físico**: diseño de la base de datos a nivel físico.

Instancias y esquemas

- **Instancia**: la información almacenada en una base de datos en un momento dado. Cambia continuamente.
- **Esquema**: diseño completo de la base de datos. Cambia poco o nada.
- **Esquema físico**: diseño de la base de datos a nivel físico.
- **Esquema lógico**: diseño de la base de datos a nivel lógico.

Instancias y esquemas

- **Instancia**: la información almacenada en una base de datos en un momento dado. Cambia continuamente.
- **Esquema**: diseño completo de la base de datos. Cambia poco o nada.
- **Esquema físico**: diseño de la base de datos a nivel físico.
- **Esquema lógico**: diseño de la base de datos a nivel lógico.
- **Sub-esquemas**: diseño de las vistas de la base de datos.

Modelos de datos

- Modelo de datos: colección de conceptos que describen/representan los datos (e.g., relaciones, restricciones)

Modelos de datos

- Modelo de datos: colección de conceptos que describen/representan los datos (e.g., relaciones, restricciones)
- Esquema: una descripción de una colección *particular* de datos, usando un modelo de datos dado

Modelos de datos

- Modelo de datos: colección de conceptos que describen/representan los datos (e.g., relaciones, restricciones)
- Esquema: una descripción de una colección *particular* de datos, usando un modelo de datos dado
- Varios modelos de datos:

Modelos de datos

- Modelo de datos: colección de conceptos que describen/representan los datos (e.g., relaciones, restricciones)
- Esquema: una descripción de una colección *particular* de datos, usando un modelo de datos dado
- Varios modelos de datos:
  - Relacional (SQLite, PostgreSQL, MySQL, MariaDB) $\rightarrow$ este curso

Modelos de datos

- Modelo de datos: colección de conceptos que describen/representan los datos (e.g., relaciones, restricciones)
- Esquema: una descripción de una colección *particular* de datos, usando un modelo de datos dado
- Varios modelos de datos:
  - Relacional (SQLite, PostgreSQL, MySQL, MariaDB) $\rightarrow$ este curso
  - NoSQL: Key-Value (Redis), Grafos (Neo4J), Documentos (MongoDB), Columnas (HBase, BigTable)

Modelos de datos

- Modelo de datos: colección de conceptos que describen/representan los datos (e.g., relaciones, restricciones)
- Esquema: una descripción de una colección *particular* de datos, usando un modelo de datos dado
- Varios modelos de datos:
  - Relacional (SQLite, PostgreSQL, MySQL, MariaDB) $\rightarrow$ este curso
  - NoSQL: Key-Value (Redis), Grafos (Neo4J), Documentos (MongoDB), Columnas (HBase, BigTable)
  - Arreglos/Matrices (ML - SciDB)

Modelos de datos

- Modelo de datos: colección de conceptos que describen/representan los datos (e.g., relaciones, restricciones)
- Esquema: una descripción de una colección *particular* de datos, usando un modelo de datos dado
- Varios modelos de datos:
  - Relacional (SQLite, PostgreSQL, MySQL, MariaDB) $\rightarrow$ este curso
  - NoSQL: Key-Value (Redis), Grafos (Neo4J), Documentos (MongoDB), Columnas (HBase, BigTable)
  - Arreglos/Matrices (ML - SciDB)
  - Jerárquicos, Red

Modelos de datos

- Modelo relacional: colección de tablas (relaciones), sus datos y las relaciones entre ellas. Modelo basado en registros. Cada tabla compuesta de registros con los mismos atributos.

Modelos de datos

- Modelo relacional: colección de tablas (relaciones), sus datos y las relaciones entre ellas. Modelo basado en registros. Cada tabla compuesta de registros con los mismos atributos.

- Modelo entidad-relación (E-R): Colección de objetos (entidades) y relaciones entre ellos. Diseño de bases de datos relacionales.

Modelos de datos

- Modelo relacional: colección de tablas (relaciones), sus datos y las relaciones entre ellas. Modelo basado en registros. Cada tabla compuesta de registros con los mismos atributos.
- Modelo entidad-relación (E-R): Colección de objetos (entidades) y relaciones entre ellos. Diseño de bases de datos relacionales.
- Modelo basado en objetos: extensión del modelo E-R para incorporar encapsulación, métodos e identidad.

Modelos de datos

- Modelo relacional: colección de tablas (relaciones), sus datos y las relaciones entre ellas. Modelo basado en registros. Cada tabla compuesta de registros con los mismos atributos.
- Modelo entidad-relación (E-R): Colección de objetos (entidades) y relaciones entre ellos. Diseño de bases de datos relacionales.
- Modelo basado en objetos: extensión del modelo E-R para incorporar encapsulación, métodos e identidad.
- Modelos semi-estructurados: objetos del mismo tipo no necesitan tener los mismos atributos.

Bases de datos
Lenguajes de una base de datos

Lenguajes de una base de datos

- DDL (data definition language): especificar el esquema de la base de datos y en general las propiedades de los datos

Lenguajes de una base de datos

- DDL (data definition language): especificar el esquema de la base de datos y en general las propiedades de los datos
- DML (data manipulation language): consultar, insertar, eliminar, modificar datos

Lenguajes de una base de datos

- DDL (data definition language): especificar el esquema de la base de datos y en general las propiedades de los datos
- DML (data manipulation language): consultar, insertar, eliminar, modificar datos
- DDL y DML en el mismo lenguaje (e.g., SQL)

Lenguajes de definición de datos

- **DDL**: data definition language

Lenguajes de definición de datos

- **DDL**: data definition language
- Restricciones sobre los datos

Lenguajes de definición de datos

- **DDL**: data definition language
- Restricciones sobre los datos
  - Restricciones de consistencia (se revisan siempre que se actualiza la base de datos)

Lenguajes de definición de datos

- **DDL**: data definition language

- Restricciones sobre los datos
  - Restricciones de consistencia (se revisan siempre que se actualiza la base de datos)
  - Restricciones de dominio: tipo y posibles valores que puede tomar un atributo (e.g., el número de créditos debe ser entero y menor a un límite que depende del tipo de matrícula del estudiante)

Lenguajes de definición de datos

- **DDL**: data definition language

- Restricciones sobre los datos
  - Restricciones de consistencia (se revisan siempre que se actualiza la base de datos)
  - Restricciones de dominio: tipo y posibles valores que puede tomar un atributo (e.g., el número de créditos debe ser entero y menor a un límite que depende del tipo de matrícula del estudiante)
  - Integridad de referencia: ciertos atributos solo pueden tomar valores definidos en otra tabla (e.g., los cursos que ve un estudiante solo pueden ser aquellos que aparezcan en la tabla de cursos)

Lenguajes de definición de datos

- Restricciones sobre los datos

Lenguajes de definición de datos

- Restricciones sobre los datos
  - Aserciones: condiciones más complejas que las anteriores (e.g., un estudiante solo puede tomar un curso si cumple con todos los pre-requisitos)

Lenguajes de definición de datos

- Restricciones sobre los datos
  - Aserciones: condiciones más complejas que las anteriores (e.g., un estudiante solo puede tomar un curso si cumple con todos los pre-requisitos)
  - Autorización: los usuarios pueden tener permisos, de lectura, inserción, actualización y/o eliminación (cualquier combinación de estos).

Lenguajes de manipulación de datos

- **DML**: data manipulation language

Lenguajes de manipulación de datos

- **DML**: data manipulation language
- Permiten almacenar y extraer información de una base de datos

Lenguajes de manipulación de datos

- **DML**: data manipulation language
- Permiten almacenar y extraer información de una base de datos
- Procedimental: la consulta especifica la estrategia (pasos de alto nivel) que debe usar el DBMS para encontrar el resultado deseado → **Álgebra Relacional**

Lenguajes de manipulación de datos

- **DML**: data manipulation language
- Permiten almacenar y extraer información de una base de datos
- Procedimental: la consulta especifica la estrategia (pasos de alto nivel) que debe usar el DBMS para encontrar el resultado deseado → **Álgebra Relacional**
- No Procedimental: la consulta especifica el resultado deseado pero no la estrategia (se determina automáticamente) → Cálculo Relacional

Bases de datos
Diseño

Diseño de bases de datos

- Determinar los requerimientos de los usuarios

Diseño de bases de datos

- Determinar los requerimientos de los usuarios
- Diseño conceptual:

Diseño de bases de datos

- Determinar los requerimientos de los usuarios
- Diseño conceptual:
  - Definir el modelo de datos a usar

Diseño de bases de datos

- Determinar los requerimientos de los usuarios
- Diseño conceptual:
  - Definir el modelo de datos a usar
  - Mapear requerimientos en el modelo de datos

Diseño de bases de datos

- Determinar los requerimientos de los usuarios
- Diseño conceptual:
  - Definir el modelo de datos a usar
  - Mapear requerimientos en el modelo de datos
  - Definir atributos y cómo agruparlos: modelo entidad-relación, normalización

Diseño de bases de datos

- Determinar los requerimientos de los usuarios
- Diseño conceptual:
  - Definir el modelo de datos a usar
  - Mapear requerimientos en el modelo de datos
  - Definir atributos y cómo agruparlos: modelo entidad-relación, normalización
  - Requerimientos funcionales: operaciones a realizar con los datos

Diseño de bases de datos

- Determinar los requerimientos de los usuarios
- Diseño conceptual

Diseño de bases de datos

- Determinar los requerimientos de los usuarios
- Diseño conceptual
- Diseño lógico: mapear el diseño conceptual (alto nivel) a una implementación del modelo de datos

Diseño de bases de datos

- Determinar los requerimientos de los usuarios
- Diseño conceptual
- Diseño lógico: mapear el diseño conceptual (alto nivel) a una implementación del modelo de datos
- Diseño físico: definir características físicas (almacenamiento)

Sobre los usuarios de la base de datos

- Tipos de usuarios:

Sobre los usuarios de la base de datos

- Tipos de usuarios:
  - Ingenuos: usan aplicaciones que se conectan a la base de datos, no lo hacen directamente

Sobre los usuarios de la base de datos

- Tipos de usuarios:
  - Ingenuos: usan aplicaciones que se conectan a la base de datos, no lo hacen directamente
  - Programadores de aplicaciones: desarrollan aplicaciones que se conectan a la base de datos

Sobre los usuarios de la base de datos

- Tipos de usuarios:
  - Ingenuos: usan aplicaciones que se conectan a la base de datos, no lo hacen directamente
  - Programadores de aplicaciones: desarrollan aplicaciones que se conectan a la base de datos
  - Sofisticados: interactúan con la base de datos sin desarrollar programas (analistas, ingenieros, científicos de datos)

Sobre los usuarios de la base de datos

- Tipos de usuarios:
  - Ingenuos: usan aplicaciones que se conectan a la base de datos, no lo hacen directamente
  - Programadores de aplicaciones: desarrollan aplicaciones que se conectan a la base de datos
  - Sofisticados: interactúan con la base de datos sin desarrollar programas (analistas, ingenieros, científicos de datos)
- Administrador de bases de datos (DBA): definición de esquemas, almacenamiento, restricciones de acceso, mantenimiento

Modelo relacional

Modelo relacional

- Estructura: relaciones (tablas) y sus contenidos

Modelo relacional

- Estructura: relaciones (tablas) y sus contenidos
- Integridad: restricciones que debe satisfacer los contenidos de la base de datos

Modelo relacional

- Estructura: relaciones (tablas) y sus contenidos
- Integridad: restricciones que debe satisfacer los contenidos de la base de datos
- Manipulación: cómo acceder y modificar la base de datos

Modelo relacional

- Estructura: relaciones (tablas) y sus contenidos
- Integridad: restricciones que debe satisfacer los contenidos de la base de datos
- Manipulación: cómo acceder y modificar la base de datos
- Ejemplo de estudiantes

Modelo relacional

- Estructura: relaciones (tablas) y sus contenidos
- Integridad: restricciones que debe satisfacer los contenidos de la base de datos
- Manipulación: cómo acceder y modificar la base de datos
- Ejemplo de estudiantes
  - Estructura: Estudiante: nombre (cadena de caracteres), apellido (cadena de caracteres), ID (long)

Modelo relacional

- Estructura: relaciones (tablas) y sus contenidos
- Integridad: restricciones que debe satisfacer los contenidos de la base de datos
- Manipulación: cómo acceder y modificar la base de datos
- Ejemplo de estudiantes
  - Estructura: Estudiante: nombre (cadena de caracteres), apellido (cadena de caracteres), ID (long)
  - Integridad: no se puede ingresar algo diferente a un long para el ID

Modelo relacional

- **Relación**: conjunto *no ordenado* que contiene los atributos que representan unas entidades

Modelo relacional

- **Relación**: conjunto *no ordenado* que contiene los atributos que representan unas entidades
- **Tupla**: conjunto de valores de los atributos en la relación (dominio)

Modelo relacional

- **Relación**: conjunto *no ordenado* que contiene los atributos que representan unas entidades

- **Tupla**: conjunto de valores de los atributos en la relación (dominio)
  - Valores atómicos/escalares (no necesariamente ahora)

Modelo relacional

- **Relación**: conjunto *no ordenado* que contiene los atributos que representan unas entidades

- **Tupla**: conjunto de valores de los atributos en la relación (dominio)
  - Valores atómicos/escalares (no necesariamente ahora)
  - El valor nulo (NULL) es miembro de todo dominio

Modelo relacional

- **Relación**: conjunto *no ordenado* que contiene los atributos que representan unas entidades
- **Tupla**: conjunto de valores de los atributos en la relación (dominio)
  - Valores atómicos/escalares (no necesariamente ahora)
  - El valor nulo (NULL) es miembro de todo dominio
- Relación con n atributos: n-aria (tabla con n columnas)

Modelo relacional - ejemplo

Ejemplo de estudiantes:

- *Relación*: {Nombre, Apellidos, ID}

Modelo relacional - ejemplo

Ejemplo de estudiantes:

- *Relación*: {Nombre, Apellidos, ID}
- *Tupla*: {Ana, Gutiérrez, 5555}

Modelo relacional - ejemplo

Ejemplo de estudiantes:

- *Relación*: {Nombre, Apellidos, ID}
- *Tupla*: {Ana, Gutiérrez, 5555}
- *Tupla*: {Juan, Pérez, 8888}

Modelo relacional - ejemplo

Ejemplo de estudiantes:

- *Relación*: {Nombre, Apellidos, ID}
- *Tupla*: {Ana, Gutiérrez, 5555}
- *Tupla*: {Juan, Pérez, 8888}

estudiantes(Nombre, Apellidos, <u>ID</u>)

Nombre	Apellidos	ID
Juan	Pérez	8888
Ana	Gutiérrez	5555

Modelo relacional

- **Llave primaria**: conjunto de atributos en una relación que identifican de manera única a cada tupla

Modelo relacional

- **Llave primaria**: conjunto de atributos en una relación que identifican de manera única a cada tupla
- Ejemplo estudiantes (ID puede ser usado como llave primaria)

Modelo relacional

- **Llave primaria**: conjunto de atributos en una relación que identifican de manera única a cada tupla
- Ejemplo estudiantes (ID puede ser usado como llave primaria)
- Ejemplo cursos: no tiene un atributo que sea llave primaria

cursos(Código, Nombre, Fecha)

Código	Nombre	Fecha
MACC01	Programación	2017-2S
MACC02	Arquitectura	2018-2S

Modelo relacional

- **Llave primaria**: conjunto de atributos en una relación que identifican de manera única a cada tupla
- Ejemplo estudiantes (ID puede ser usado como llave primaria)
- Ejemplo cursos: no tiene un atributo que sea llave primaria
- Se debe generar una llave primaria (algunos DMBS lo hacen automáticamente)

cursos(id, Código, Nombre, Fecha)

id	Código	Nombre	Fecha
000	MACC01	Programación	2017-2S
001	MACC02	Arquitectura	2018-2S

Modelo relacional

- **Llave primaria**: conjunto de atributos en una relación que identifican de manera única a cada tupla
- Ejemplo estudiantes (ID puede ser usado como llave primaria)
- Ejemplo cursos: no tiene un atributo que sea llave primaria
- Se debe generar una llave primaria (algunos DMBS lo hacen automáticamente)
- PosgreSQL: SERIAL - MySQL: AUTO_INCREMENT

cursos(<u>id</u>, Código, Nombre, Fecha)

id	Código	Nombre	Fecha
000	MACC01	Programación	2017-2S
001	MACC02	Arquitectura	2018-2S

Modelo relacional

- **Llave foránea**: especifica la asociación de un atributo de una relación con un atributo en *otra* relación

Modelo relacional

- **Llave foránea**: especifica la asociación de un atributo de una relación con un atributo en *otra* relación
- Permite relacionar elementos en diferentes tablas y mantenerlos sincronizados

Modelo relacional

- **Llave foránea**: especifica la asociación de un atributo de una relación con un atributo en *otra* relación
- Permite relacionar elementos en diferentes tablas y mantenerlos sincronizados
- Ejemplo: queremos agregar estudiantes a cursos

estudiantes(Nombre, Apellidos, ID)

Nombre	Apellidos	ID
Juan	Pérez	8888
Ana	Gutiérrez	5555

cursos(id, Código, Nombre, Fecha)

id	Código	Nombre	Fecha
000	MACC01	Programación	2017-2S
001	MACC02	Arquitectura	2018-2S

Llave foránea

Ejemplo: queremos agregar estudiantes a cursos

- Nueva tabla *estudiantesCursos* relaciona estudiantes con cursos

Llave foránea

Ejemplo: queremos agregar estudiantes a cursos

- Nueva tabla *estudiantesCursos* relaciona estudiantes con cursos

estudiantesCursos(estudianteID, cursoID)

estudianteID	cursoID
8888	000
5555	000
5555	001

Llave foránea

Ejemplo: queremos agregar estudiantes a cursos

- Nueva tabla *estudiantesCursos* relaciona estudiantes con cursos

estudiantesCursos(estudianteID, cursoID)

estudianteID	cursoID
8888	000
5555	000
5555	001

estudiantesCursos(estudianteID, cursoID, relID)

estudianteID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

Llave foránea

- Nueva tabla *estudiantesCursos* relaciona estudiantes con cursos (referencia cruzada)

estudiantesCursos(estudianteID, cursoID, relID)

estudianteID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

Llave foránea

- Nueva tabla *estudiantesCursos* relaciona estudiantes con cursos (referencia cruzada)
- Llave foránea: el atributo cursoID de la tabla *estudiantesCursos* se relaciona con el atributo id de la tabla *cursos*

estudiantesCursos(estudianteID, cursoID, relID)

estudianteID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

cursos(id, Código, Nombre, Fecha)

id	Código	Nombre	Fecha
000	MACC01	Programación	2017-2S
001	MACC02	Arquitectura	2018-2S

Llave foránea

- Nueva tabla *estudiantesCursos* relaciona estudiantes con cursos (referencia cruzada)

estudiantesCursos(estudianteID, cursoID, relID)

estudianteID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

Llave foránea

- Nueva tabla *estudiantesCursos* relaciona estudiantes con cursos (referencia cruzada)
- Llave foránea: el atributo estudianteID de la tabla *estudiantesCursos* se relaciona con el atributo ID de la tabla *estudiantes*

estudiantesCursos(estudianteID, cursoID, relID)

estudianteID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

estudiantes(Nombre, Apellidos, ID)

Nombre	Apellidos	ID
Juan	Pérez	8888
Ana	Gutiérrez	5555

Llave foránea

- Si se elimina un registro en la tabla estudiantes, se eliminan todos los registros en la tabla estudiantesCursos

Llave foránea

- Si se elimina un registro en la tabla estudiantes, se eliminan todos los registros en la tabla estudiantesCursos
- Al agregar un registro en la tabla estudiantesCursos se verifica que tanto el estudiante como el curso *existan* en las tablas correspondientes

Llave foránea

- Si se elimina un registro en la tabla estudiantes, se eliminan todos los registros en la tabla estudiantesCursos
- Al agregar un registro en la tabla estudiantesCursos se verifica que tanto el estudiante como el curso *existan* en las tablas correspondientes
- El DMBS se encarga de esto (una vez se definen adecuadamente las llaves foráneas)

Álgebra relacional

Álgebra relacional

- Operaciones para manipular las tuplas en una relación

Álgebra relacional

- Operaciones para manipular las tuplas en una relación
- Álgebra de conjuntos

Álgebra relacional

- Operaciones para manipular las tuplas en una relación
- Álgebra de conjuntos
- 7 operadores:
  Selección σ, Proyección π, Unión $\cup$, Intersección $\cap$, Diferencia $-$,
  Producto $\times$, Join $\bowtie$

Álgebra relacional

- Operaciones para manipular las tuplas en una relación
- Álgebra de conjuntos
- 7 operadores:
  Selección σ, Proyección π, Unión $\cup$, Intersección $\cap$, Diferencia $-$,
  Producto $\times$, Join $\bowtie$
- Operador:
  - Entrada: una o más relaciones
  - Salida: una nueva relación

Álgebra relacional

- Operaciones para manipular las tuplas en una relación
- Álgebra de conjuntos
- 7 operadores:
  Selección σ, Proyección π, Unión $\cup$, Intersección $\cap$, Diferencia $-$,
  Producto $\times$, Join $\bowtie$
- Operador:
  - Entrada: una o más relaciones
  - Salida: una nueva relación
- Operadores se pueden encadenar

Álgebra relacional: seleccionar σ

- Seleccionar un subconjunto de las tuplas de una relación que satisfacen un predicado de selección

Álgebra relacional: seleccionar σ

- Seleccionar un subconjunto de las tuplas de una relación que satisfacen un predicado de selección
- Predicado sirve para filtrar las tuplas, manteniendo solo las que cumplan con la condición

Álgebra relacional: seleccionar σ

- Seleccionar un subconjunto de las tuplas de una relación que satisfacen un predicado de selección
- Predicado sirve para filtrar las tuplas, manteniendo solo las que cumplan con la condición
- El predicado puede combinar varias condiciones (lógica booleana)

Álgebra relacional: seleccionar σ

- Seleccionar un subconjunto de las tuplas de una relación que satisfacen un predicado de selección
- Predicado sirve para filtrar las tuplas, manteniendo solo las que cumplan con la condición
- El predicado puede combinar varias condiciones (lógica booleana)
- $\sigma_{\text{predicado}}(\text{Relación})$

Álgebra relacional: seleccionar σ

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

Álgebra relacional: seleccionar σ

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

$\sigma_{\text{cursoID}='000'}(\text{estCursos})$

estID	cursoID	relID
8888	000	1000
5555	000	1001

Álgebra relacional: seleccionar σ

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

$\sigma_{\text{cursoID}='000'}(\text{estCursos})$

estID	cursoID	relID
8888	000	1000
5555	000	1001

```
SELECT * FROM estCursos WHERE cursoID='000';
```

Álgebra relacional: seleccionar σ

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

Álgebra relacional: seleccionar σ

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

$\sigma_{\text{cursoID='000'} \ \wedge \ \text{estID='5555'}}(\text{estCursos})$

estID	cursoID	relID
5555	000	1001

Álgebra relacional: seleccionar σ

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

$\sigma_{\text{cursoID='000'} \ \wedge \ \text{estID='5555'}}(\text{estCursos})$

estID	cursoID	relID
5555	000	1001

```
SELECT * FROM estCursos WHERE cursoID='000' and estID='5555'
    ;
```

Álgebra relacional: proyección π

- Generar una nueva relación que contenga solo un subconjunto de los atributos de una relación (y todas sus tuplas)

Álgebra relacional: proyección π

- Generar una nueva relación que contenga solo un subconjunto de los atributos de una relación (y todas sus tuplas)
- Se puede modificar el valor de cada atributo (para todas las tuplas)

Álgebra relacional: proyección π

- Generar una nueva relación que contenga solo un subconjunto de los atributos de una relación (y todas sus tuplas)
- Se puede modificar el valor de cada atributo (para todas las tuplas)
- Se puede modificar el orden de los atributos

Álgebra relacional: proyección π

- Generar una nueva relación que contenga solo un subconjunto de los atributos de una relación (y todas sus tuplas)
- Se puede modificar el valor de cada atributo (para todas las tuplas)
- Se puede modificar el orden de los atributos
- $\pi_{A_1,\ldots,A_n}(\text{Relación})$

Álgebra relacional: proyección π

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

Álgebra relacional: proyección π

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

$\pi_{\text{cursoID}+1,estID}(\text{estCursos})$

cursoID	estID
001	8888
001	5555
002	5555

Álgebra relacional: proyección π

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

$\pi_{\text{cursoID}+1,estID}(\text{estCursos})$

cursoID	estID
001	8888
001	5555
002	5555

```
SELECT cursoID+1, estID FROM estCursos;
```

Álgebra relacional: proyección π

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

Álgebra relacional: proyección π

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

$\pi_{\text{cursoID}+1,estID}(\sigma_{\text{cursoID}='000'}(\text{estCursos}))$

cursoID	estID
001	8888
001	5555

Álgebra relacional: proyección π

estCursos(estID, cursoID, relID)

estID	cursoID	relID
8888	000	1000
5555	000	1001
5555	001	1002

$\pi_{\text{cursoID}+1,estID}(\sigma_{\text{cursoID}='000'}(\text{estCursos}))$

cursoID	estID
001	8888
001	5555

```
SELECT cursoID+1, estID FROM estCursos WHERE cursoID='000';
```

Álgebra relacional: unión $\cup$

- Generar una nueva relación que contiene todas las tuplas que aparecen en al menos una de las relaciones de entrada

Álgebra relacional: unión $\cup$

- Generar una nueva relación que contiene todas las tuplas que aparecen en al menos una de las relaciones de entrada
- Las relaciones de entrada deben tener los mismos atributos

Álgebra relacional: unión $\cup$

- Generar una nueva relación que contiene todas las tuplas que aparecen en al menos una de las relaciones de entrada
- Las relaciones de entrada deben tener los mismos atributos
- $\text{Relación}_1 \cup \text{Relación}_2$

Álgebra relacional: unión $\cup$

cursos1(Código, Nombre, Fecha)

Código	Nombre	Fecha
MACC01	Programación	2017-2S
MACC02	Arquitectura	2018-2S

cursos3(Código, Nombre, Fecha)

Código	Nombre	Fecha
MACC10	Probabilidad	2018-2S
MACC11	Grafos	2018-2S
MACC02	Arquitectura	2018-2S

Álgebra relacional: unión $\cup$

cursos1$\cup$cursos3

Código	Nombre	Fecha
MACC01	Programación	2017-2S
MACC02	Arquitectura	2018-2S
MACC10	Probabilidad	2018-2S
MACC11	Grafos	2018-2S

Álgebra relacional: unión $\cup$

cursos1$\cup$cursos3

Código	Nombre	Fecha
MACC01	Programación	2017-2S
MACC02	Arquitectura	2018-2S
MACC10	Probabilidad	2018-2S
MACC11	Grafos	2018-2S

```
(SELECT * FROM cursos1) UNION (SELECT * FROM cursos3);
```

Álgebra relacional: intersección $\cap$

- Generar una nueva relación que contiene las tuplas que aparecen en todas las relaciones de entrada

Álgebra relacional: intersección $\cap$

- Generar una nueva relación que contiene las tuplas que aparecen en todas las relaciones de entrada
- Las relaciones de entrada deben tener los mismos atributos

Álgebra relacional: intersección $\cap$

- Generar una nueva relación que contiene las tuplas que aparecen en todas las relaciones de entrada
- Las relaciones de entrada deben tener los mismos atributos
- $\text{Relación}_1 \cap \text{Relación}_2$

Álgebra relacional: intersección ∩

cursos1(Código, Nombre, Fecha)

Código	Nombre	Fecha
MACC01	Programación	2017-2S
MACC02	Arquitectura	2018-2S

cursos3(Código, Nombre, Fecha)

Código	Nombre	Fecha
MACC10	Probabilidad	2018-2S
MACC11	Grafos	2018-2S
MACC02	Arquitectura	2018-2S

Álgebra relacional: intersección ∩

cursos1∩cursos3

Código	Nombre	Fecha
MACC02	Arquitectura	2018-2S

Álgebra relacional: intersección ∩

cursos1∩cursos3

Código	Nombre	Fecha
MACC02	Arquitectura	2018-2S

```
(SELECT * FROM cursos1) INTERSECT (SELECT * FROM cursos3);
```

Álgebra relacional: diferencia −

- Generar una nueva relación que contiene las tuplas que aparecen en la primera relación de entrada pero no en la segunda

Álgebra relacional: diferencia −

- Generar una nueva relación que contiene las tuplas que aparecen en la primera relación de entrada pero no en la segunda
- Las relaciones de entrada deben tener los mismos atributos

Álgebra relacional: diferencia $-$

- Generar una nueva relación que contiene las tuplas que aparecen en la primera relación de entrada pero no en la segunda
- Las relaciones de entrada deben tener los mismos atributos
- $\text{Relación}_1 - \text{Relación}_2$

Álgebra relacional: diferencia −

cursos1(Código, Nombre, Fecha)

Código	Nombre	Fecha
MACC01	Programación	2017-2S
MACC02	Arquitectura	2018-2S

cursos3(Código, Nombre, Fecha)

Código	Nombre	Fecha
MACC10	Probabilidad	2018-2S
MACC11	Grafos	2018-2S
MACC02	Arquitectura	2018-2S

Álgebra relacional: diferencia −

cursos1 − cursos3

Código	Nombre	Fecha
MACC01	Programación	2017-2S

Álgebra relacional: diferencia −

cursos1 − cursos3

Código	Nombre	Fecha
MACC01	Programación	2017-2S

```
(SELECT * FROM cursos1) EXCEPT (SELECT * FROM cursos3);
```

Álgebra relacional: producto $\times$

- Generar una nueva relación que contiene todas las combinaciones posibles de las tuplas en las relaciones de entrada

Álgebra relacional: producto $\times$

- Generar una nueva relación que contiene todas las combinaciones posibles de las tuplas en las relaciones de entrada
- Conserva todos los atributos de las relaciones de entrada

Álgebra relacional: producto $\times$

- Generar una nueva relación que contiene todas las combinaciones posibles de las tuplas en las relaciones de entrada
- Conserva todos los atributos de las relaciones de entrada
- $\text{Relación}_1 \times \text{Relación}_2$

Álgebra relacional: producto $\times$

estudiantes(Nombre, Apellidos, ID)

Nombre	Apellidos	ID
Juan	Pérez	8888
Ana	Gutiérrez	5555

cursos(Código, Nombre, Fecha)

Código	Nombre	Fecha
MACC01	Programación	2017-2S
MACC02	Arquitectura	2018-2S

Álgebra relacional: producto $\times$

cursos $\times$ estudiantes

Código	NombreC	Fecha	NombreE	Apellidos	ID
MACC01	Programación	2017-2S	Juan	Pérez	8888
MACC01	Programación	2017-2S	Ana	Gutiérrez	5555
MACC02	Arquitectura	2018-2S	Juan	Pérez	8888
MACC02	Arquitectura	2018-2S	Ana	Gutiérrez	5555

Álgebra relacional: producto $\times$

cursos $\times$ estudiantes

Código	NombreC	Fecha	NombreE	Apellidos	ID
MACC01	Programación	2017-2S	Juan	Pérez	8888
MACC01	Programación	2017-2S	Ana	Gutiérrez	5555
MACC02	Arquitectura	2018-2S	Juan	Pérez	8888
MACC02	Arquitectura	2018-2S	Ana	Gutiérrez	5555

```
SELECT * FROM cursos CROSS JOIN estudiantes;
```

Álgebra relacional: producto $\times$

cursos $\times$ estudiantes

Código	NombreC	Fecha	NombreE	Apellidos	ID
MACC01	Programación	2017-2S	Juan	Pérez	8888
MACC01	Programación	2017-2S	Ana	Gutiérrez	5555
MACC02	Arquitectura	2018-2S	Juan	Pérez	8888
MACC02	Arquitectura	2018-2S	Ana	Gutiérrez	5555

```
SELECT * FROM cursos CROSS JOIN estudiantes;
```

```
SELECT * FROM cursos, estudiantes;
```

Álgebra relacional: join $\bowtie$

- Generar una nueva relación que contiene todas las combinaciones posibles de las tuplas en las relaciones de entrada *siempre que las tuplas consideradas en una combinación coincidan en el valor de sus atributos comunes*

Álgebra relacional: join $\bowtie$

- Generar una nueva relación que contiene todas las combinaciones posibles de las tuplas en las relaciones de entrada *siempre que las tuplas consideradas en una combinación coincidan en el valor de sus atributos comunes*

- El nombre de los atributos en que coinciden deben ser iguales

Álgebra relacional: join $\bowtie$

- Generar una nueva relación que contiene todas las combinaciones posibles de las tuplas en las relaciones de entrada *siempre que las tuplas consideradas en una combinación coincidan en el valor de sus atributos comunes*
- El nombre de los atributos en que coinciden deben ser iguales
- Conserva todos los atributos de las dos relaciones

Álgebra relacional: join $\bowtie$

- Generar una nueva relación que contiene todas las combinaciones posibles de las tuplas en las relaciones de entrada *siempre que las tuplas consideradas en una combinación coincidan en el valor de sus atributos comunes*
- El nombre de los atributos en que coinciden deben ser iguales
- Conserva todos los atributos de las dos relaciones
- Diferente de $\cup$ donde los atributos de las dos relaciones deben ser iguales

Álgebra relacional: join $\bowtie$

- Generar una nueva relación que contiene todas las combinaciones posibles de las tuplas en las relaciones de entrada *siempre que las tuplas consideradas en una combinación coincidan en el valor de sus atributos comunes*
- El nombre de los atributos en que coinciden deben ser iguales
- Conserva todos los atributos de las dos relaciones
- Diferente de $\cup$ donde los atributos de las dos relaciones deben ser iguales
- $\text{Relación}_1 \bowtie \text{Relación}_2$

Álgebra relacional: join $\bowtie$

cursos(Código, Asignatura, Fecha)

Código	Asignatura	Fecha
MACC10	Probabilidad	2018-2S
MACC11	Grafos	2018-2S
MACC02	Arquitectura	2018-2S
MACC12	Programación	2017-2S

libros(Asignatura, Libro, Autor)

Asignatura	Libro	Autor
Programación	The Art of Programming	E. Roberts
Probabilidad	Intro to probability	D. Bertsekas

Álgebra relacional: join ⋈

cursos ⋈ libros

Código	Nombre	Fecha	Libro	Autor
MACC10	Probabilidad	2018-2S	Intro to probability	D. Bertsekas
MACC12	Programación	2017-2S	The Art of Programming	E. Roberts

Álgebra relacional: join ⋈

cursos ⋈ libros

Código	Nombre	Fecha	Libro	Autor
MACC10	Probabilidad	2018-2S	Intro to probability	D. Bertsekas
MACC12	Programación	2017-2S	The Art of Programming	E. Roberts

```
SELECT * FROM cursos NATURAL JOIN libros;
```

Álgebra relacional: otros operadores

- Renombrar ρ

Álgebra relacional: otros operadores

- Renombrar ρ
- Asignación $\leftarrow$

Álgebra relacional: otros operadores

- Renombrar ρ
- Asignación $\leftarrow$
- Eliminar duplicados δ

Álgebra relacional: otros operadores

- Renombrar ρ
- Asignación $\leftarrow$
- Eliminar duplicados δ
- Agregar γ

Álgebra relacional: otros operadores

- Renombrar ρ
- Asignación $\leftarrow$
- Eliminar duplicados δ
- Agregar γ
- Ordenamiento τ

Álgebra relacional: otros operadores

- Renombrar ρ
- Asignación $\leftarrow$
- Eliminar duplicados δ
- Agregar γ
- Ordenamiento τ
- División $\div$

Álgebra relacional vs. SQL

Álgebra relacional vs. SQL

- SQL: structured query language

Álgebra relacional vs. SQL

- SQL: structured query language
- Álgebra relacional define los pasos que deben seguirse para ejecutar la consulta

Álgebra relacional vs. SQL

- SQL: structured query language
- Álgebra relacional define los pasos que deben seguirse para ejecutar la consulta
- Ejemplo:

$$\pi_{\text{cursoID}+1,estID}(\sigma_{\text{cursoID}='000'}(\text{estCursos}))$$

Álgebra relacional vs. SQL

- SQL: structured query language
- Álgebra relacional define los pasos que deben seguirse para ejecutar la consulta
- Ejemplo:

$$\pi_{\text{cursoID}+1,estID}(\sigma_{\text{cursoID}='000'}(\text{estCursos}))$$

- SQL no define los pasos

Álgebra relacional vs. SQL

- SQL: structured query language
- Álgebra relacional define los pasos que deben seguirse para ejecutar la consulta
- Ejemplo:

$$\pi_{\text{cursoID}+1,estID}(\sigma_{\text{cursoID}='000'}(\text{estCursos}))$$

- SQL no define los pasos
- Ejemplo:

```
SELECT cursoID+1, estID FROM estCursos WHERE cursoID='
    000';
```

Álgebra relacional vs. SQL

- SQL define cómo ejecutar los pasos, no es necesario especificar el orden

Álgebra relacional vs. SQL

- SQL define cómo ejecutar los pasos, no es necesario especificar el orden
- Optimización de consultas: ¿mejor forma de ejecutar una consulta para minimizar el tiempo de ejecución?

Álgebra relacional vs. SQL

- SQL define cómo ejecutar los pasos, no es necesario especificar el orden
- Optimización de consultas: ¿mejor forma de ejecutar una consulta para minimizar el tiempo de ejecución?
- Separación de la definición de la consulta y cómo debe ejecutarse a alto nivel (además de la separación de ejecución a bajo nivel)

Álgebra relacional vs. SQL

- SQL define cómo ejecutar los pasos, no es necesario especificar el orden
- Optimización de consultas: ¿mejor forma de ejecutar una consulta para minimizar el tiempo de ejecución?
- Separación de la definición de la consulta y cómo debe ejecutarse a alto nivel (además de la separación de ejecución a bajo nivel)

```
SELECT cursoID+1, estID FROM estCursos WHERE cursoID='
    000';
```

Resumen

Resumen

- Bases de datos: requerimientos, DBMS, etc.
- Modelos de datos
- Lenguajes de definición y manipulación de datos
- Diseño de bases de datos
- Modelo de datos relacional
- Álgebra relacional: operadores
- SQL: structure query language

Próximamente

- SQL básico
- SQL intermedio
- SQL avanzado