The Relational Database Model

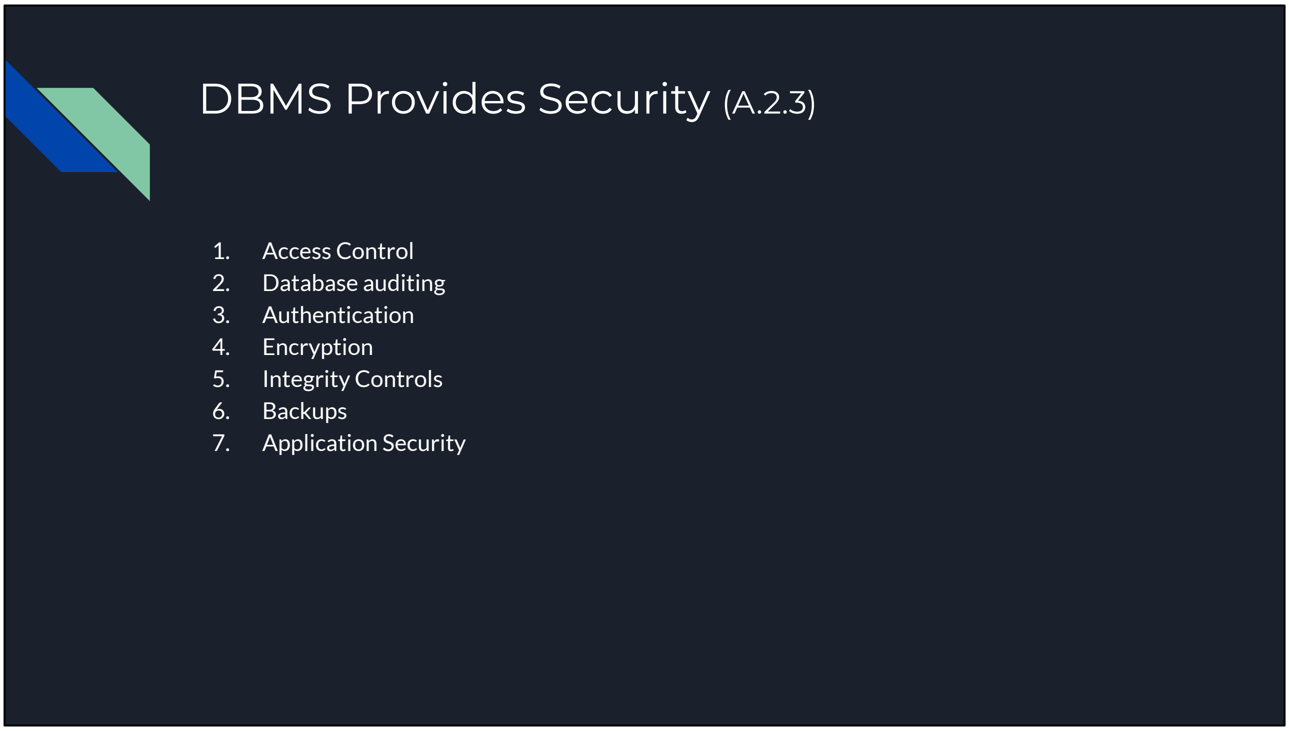
(A.2)

(R)DBMS (Relational Database Management System) (A.2.1)

1. A software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure. It also defines rules to validate and manipulate this data.
2. A database management system receives instruction from a database administrator (DBA) and accordingly instructs the system to make the necessary changes. These commands can be to load, retrieve or modify existing data from the system.
3. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

DBMS Functions (A.2.2)

- Provides security
- Self-describing nature of a database system
- Insulation between programs and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing
- Control of data redundancy
- Data sharing
- Enforcement of integrity constraints
- Restriction of unauthorized access
- Data independence
- Transaction processing
- Backup and recovery facilities

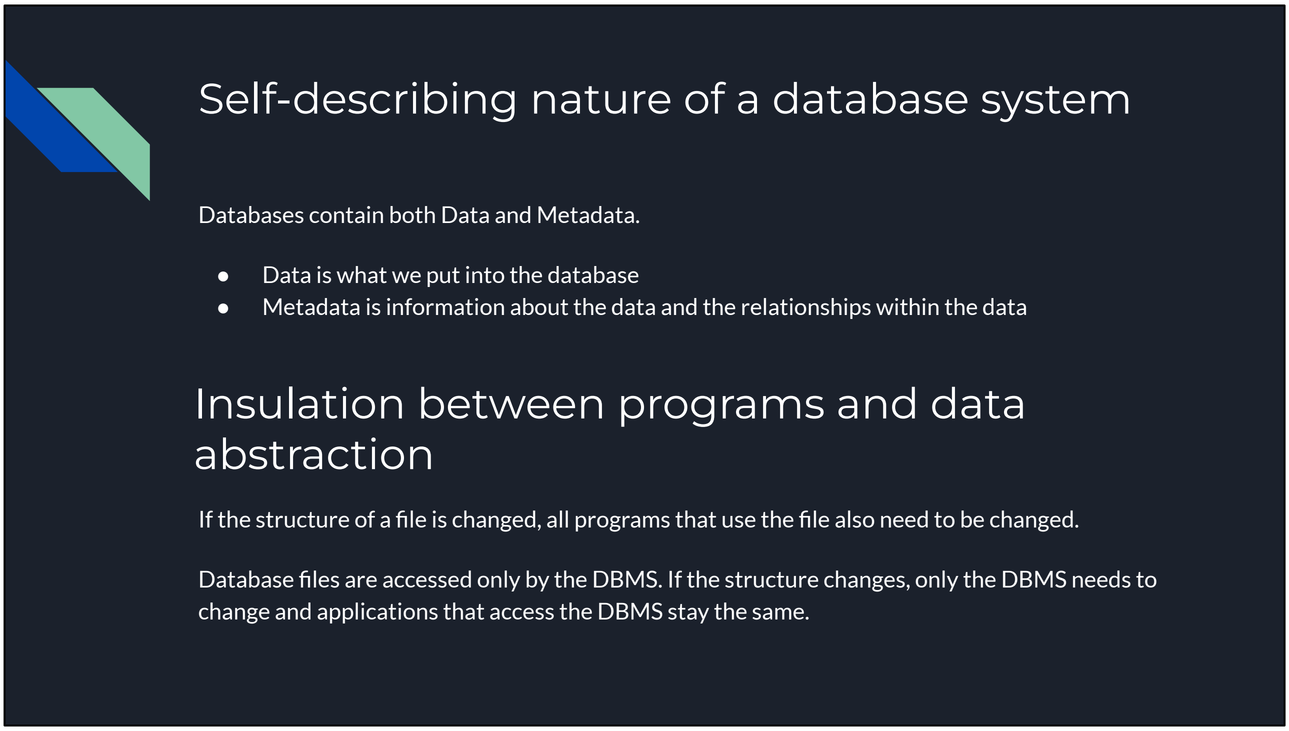


DBMS Provides Security (A.2.3)

1. Access Control
2. Database auditing
3. Authentication
4. Encryption
5. Integrity Controls
6. Backups
7. Application Security

1. In the fields of physical security and information security, access control (AC) is the selective restriction of access to a place or other resource. The act of accessing may mean consuming, entering, or using. Permission to access a resource is called authorization.
2. Database auditing involves observing a database so as to be aware of the actions of database users. Database administrators and consultants often set up auditing for security purposes, for example, to ensure that those without the permission to access information do not access it.
3. Authentication is the act of proving an assertion, such as the identity of a computer system user.
4. In cryptography, encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot.
5. Data integrity is the maintenance of, and the assurance of the accuracy and consistency of, data over its entire life-cycle,[1] and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data. The term is broad in scope and may have widely different meanings depending on the specific context – even under the same general umbrella of computing. It is at times used as a proxy term for data quality,[2] while data validation is a pre-requisite for data integrity.[3] Data integrity is the opposite of data corruption.[4] The overall intent of any data integrity technique is the same: ensure data is recorded exactly as intended (such as a database correctly rejecting mutually exclusive possibilities,) and upon later retrieval, ensure the data is the same as it was when it was originally recorded. In short,

1. data integrity aims to prevent unintentional changes to information. Data integrity is not to be confused with data security, the discipline of protecting data from unauthorized parties.
2. In information technology, a backup, or data backup is a copy of computer data taken and stored elsewhere so that it may be used to restore the original after a data loss event.
3. Application security encompasses measures taken to improve the security of an application often by finding, fixing and preventing security vulnerabilities.



Self-describing nature of a database system

Databases contain both Data and Metadata.

- Data is what we put into the database
- Metadata is information about the data and the relationships within the data

Insulation between programs and data abstraction

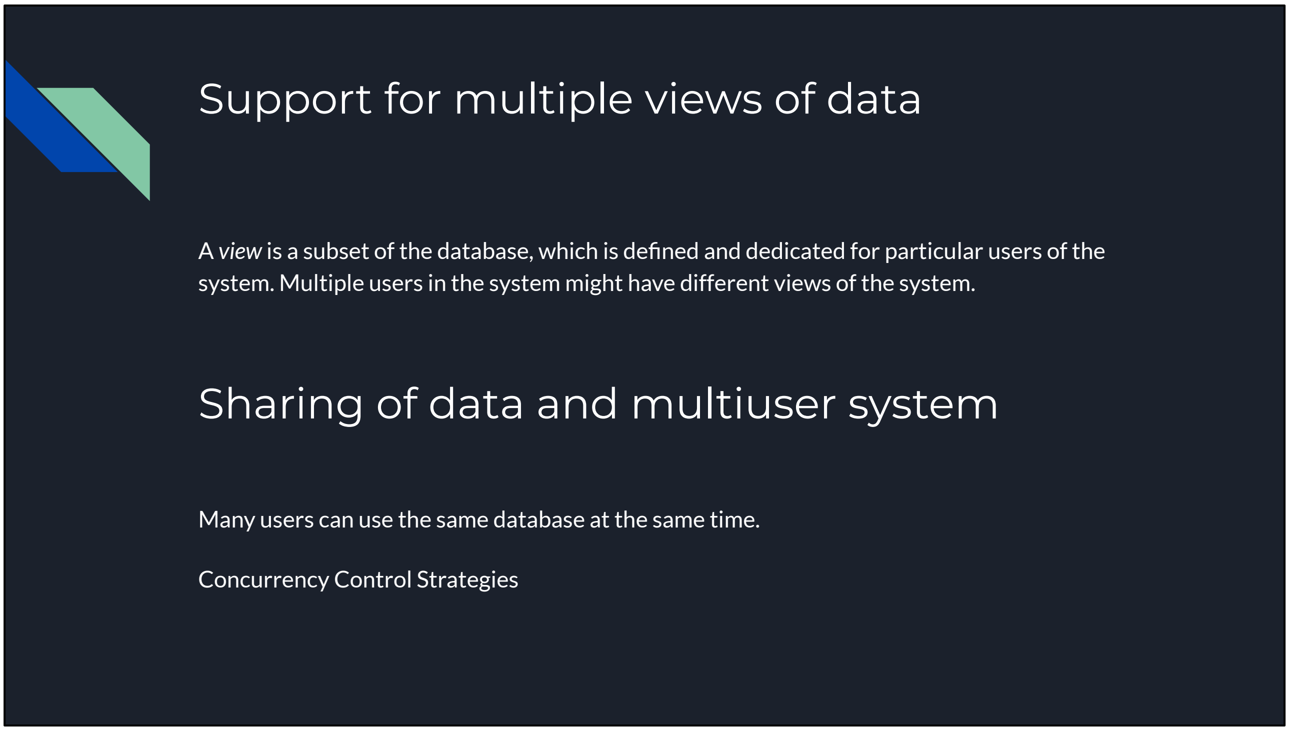
If the structure of a file is changed, all programs that use the file also need to be changed.

Database files are accessed only by the DBMS. If the structure changes, only the DBMS needs to change and applications that access the DBMS stay the same.

Self-describing nature of a database system
A database system is referred to as self-describing because it not only contains the database itself, but also metadata which defines and describes the data and relationships between tables in the database. This information is used by the DBMS software or database users if needed. This separation of data and information about the data makes a database system totally different from the traditional file-based system in which the data definition is part of the application programs.

Insulation between program and data
In the file-based system, the structure of the data files is defined in the application programs so if a user wants to change the structure of a file, all the programs that access that file might need to be changed as well.

On the other hand, in the database approach, the data structure is stored in the system catalogue and not in the programs. Therefore, one change is all that is needed to change the structure of a file. This insulation between the programs and data is also called program-data independence.



Support for multiple views of data

A *view* is a subset of the database, which is defined and dedicated for particular users of the system. Multiple users in the system might have different views of the system.

Sharing of data and multiuser system

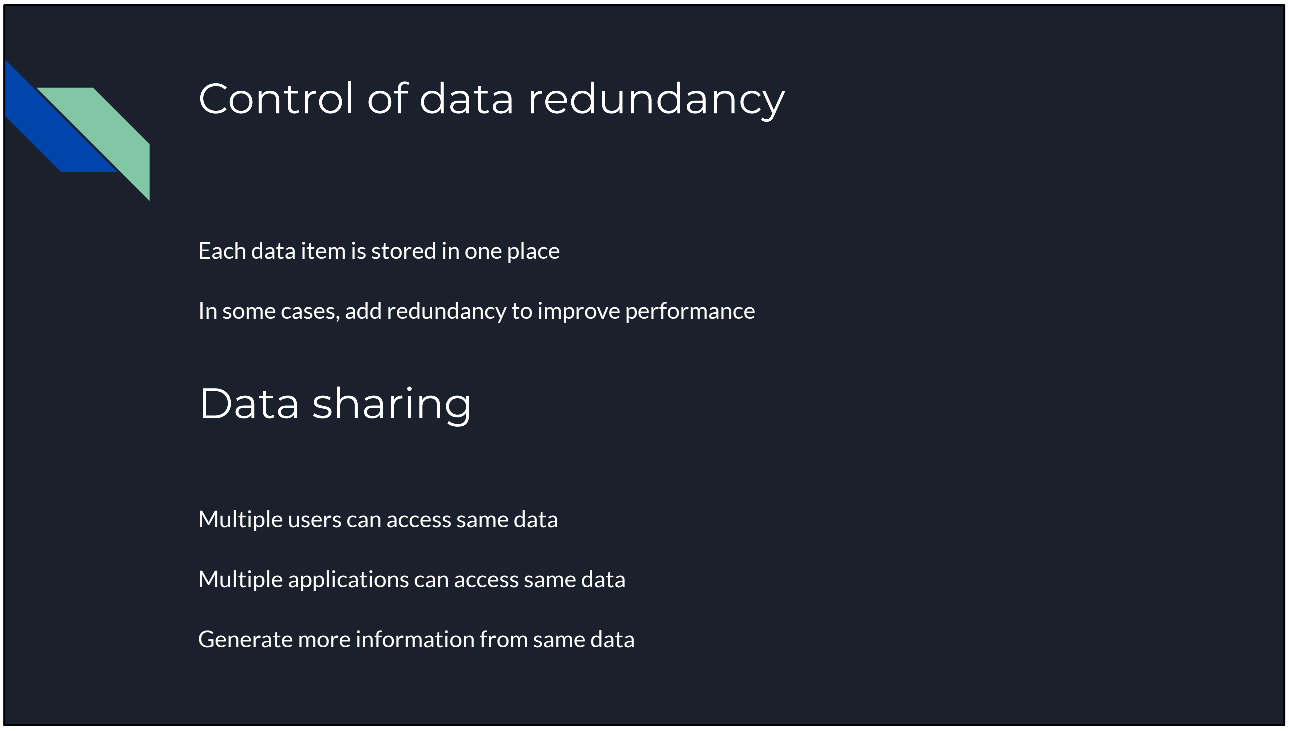
Many users can use the same database at the same time.

Concurrency Control Strategies

Support for multiple views of data
A database supports multiple views of data. A view is a subset of the database, which is defined and dedicated for particular users of the system. Multiple users in the system might have different views of the system. Each view might contain only the data of interest to a user or group of users.

Sharing of data and multiuser system
Current database systems are designed for multiple users. That is, they allow many users to access the same database at the same time. This access is achieved through features called concurrency control strategies. These strategies ensure that the data accessed are always correct and that data integrity is maintained.

The design of modern multiuser database systems is a great improvement from those in the past which restricted usage to one person at a time.


Control of data redundancy

Each data item is stored in one place

In some cases, add redundancy to improve performance

Data sharing

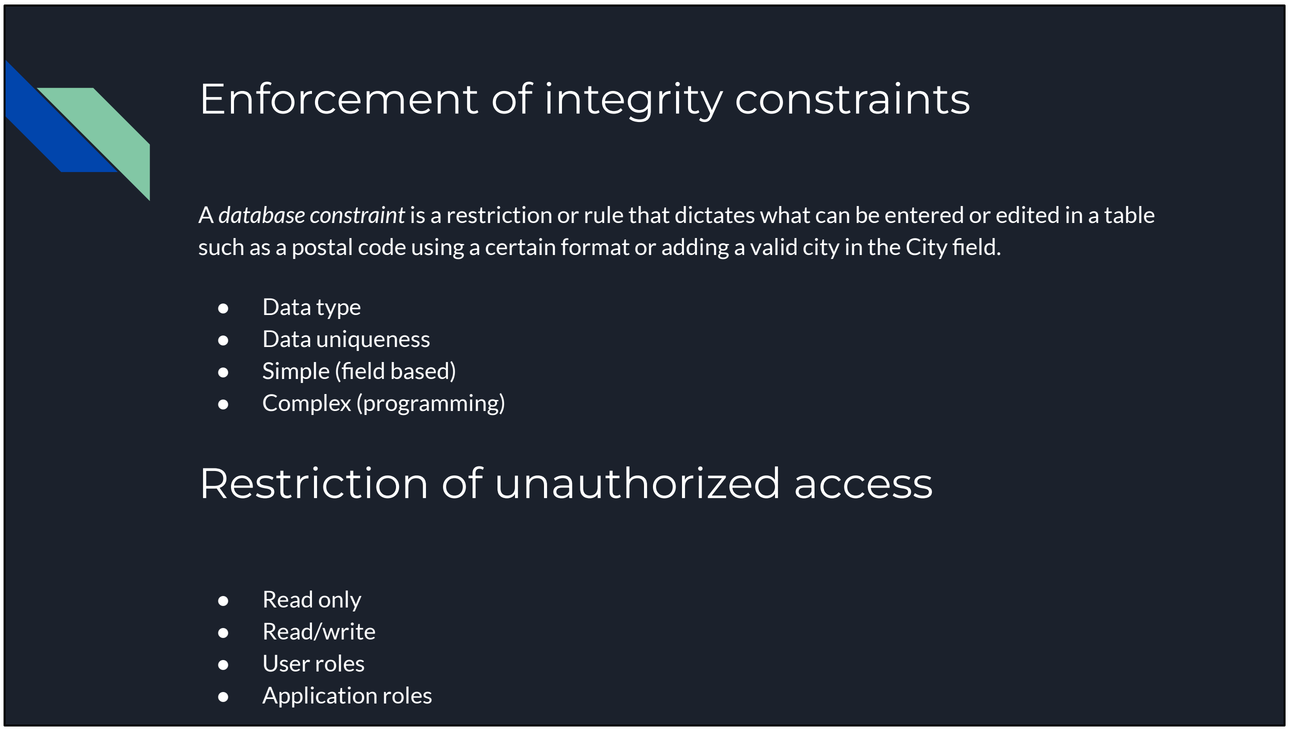
Multiple users can access same data

Multiple applications can access same data

Generate more information from same data

Control of data redundancy
In the database approach, ideally, each data item is stored in only one place in the database. In some cases, data redundancy still exists to improve system performance, but such redundancy is controlled by application programming and kept to minimum by introducing as little redudancy as possible when designing the database.

Data sharing
The integration of all the data, for an organization, within a database system has many advantages. First, it allows for data sharing among employees and others who have access to the system. Second, it gives users the ability to generate more information from a given amount of data than would be possible without the integration.


Enforcement of integrity constraints

A *database constraint* is a restriction or rule that dictates what can be entered or edited in a table such as a postal code using a certain format or adding a valid city in the City field.

- Data type
- Data uniqueness
- Simple (field based)
- Complex (programming)

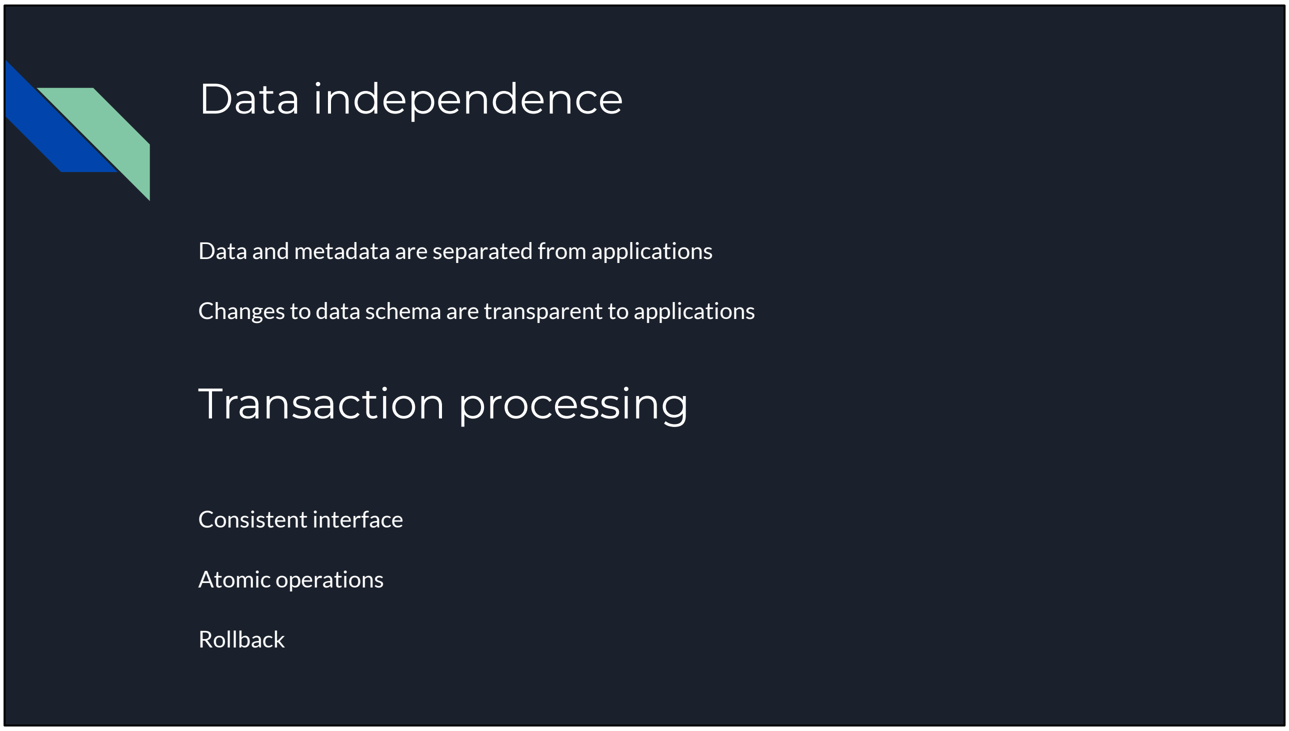
Restriction of unauthorized access

- Read only
- Read/write
- User roles
- Application roles

Enforcement of integrity constraints
Database management systems must provide the ability to define and enforce certain constraints to ensure that users enter valid information and maintain data integrity. A database constraint is a restriction or rule that dictates what can be entered or edited in a table such as a postal code using a certain format or adding a valid city in the City field.

There are many types of database constraints. Data type, for example, determines the sort of data permitted in a field, for example numbers only. Data uniqueness such as the primary key ensures that no duplicates are entered. Constraints can be simple (field based) or complex (programming).

Restriction of unauthorized access
Not all users of a database system will have the same accessing privileges. For example, one user might have read-only access (i.e., the ability to read a file but not make changes), while another might have read and write privileges, which is the ability to both read and modify a file. For this reason, a database management system should provide a security subsystem to create and control different types of user accounts and restrict unauthorized access.



Data independence

Data and metadata are separated from applications

Changes to data schema are transparent to applications

Transaction processing

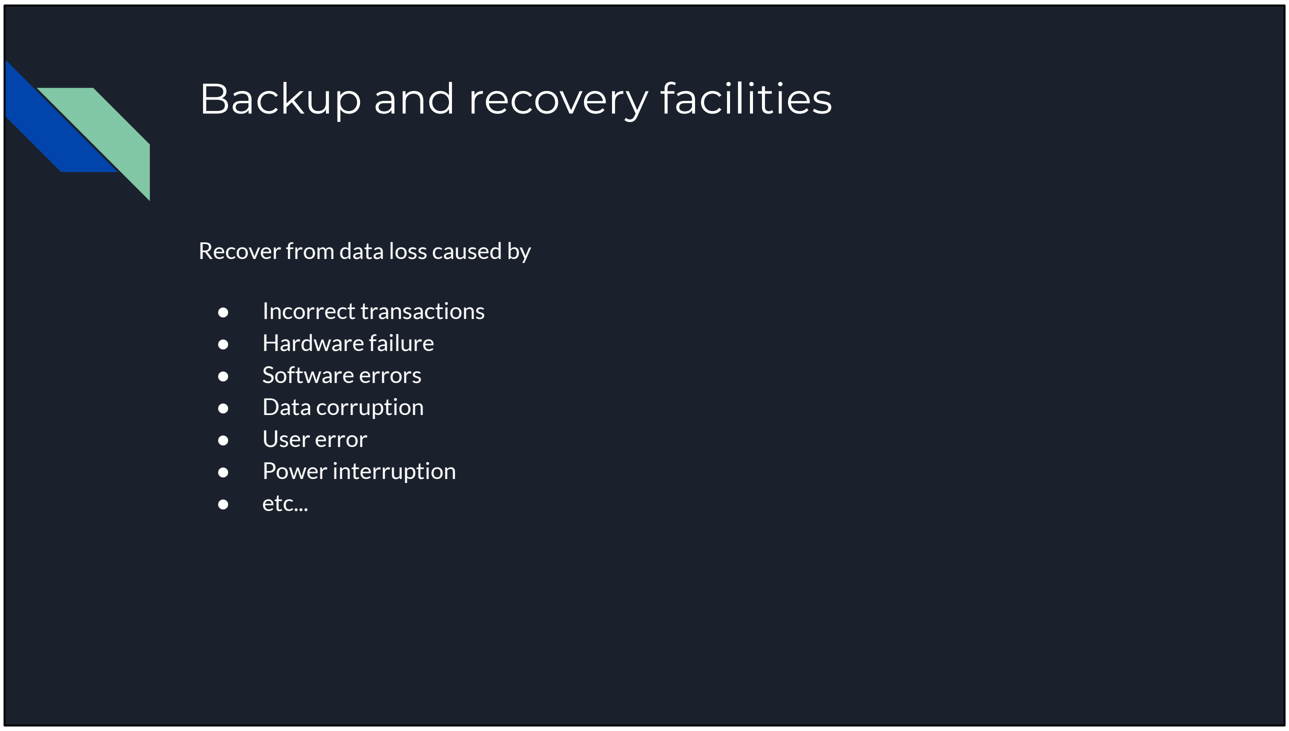
Consistent interface

Atomic operations

Rollback

Data independence
Another advantage of a database management system is how it allows for data independence. In other words, the system data descriptions or data describing data (metadata) are separated from the application programs. This is possible because changes to the data structure are handled by the database management system and are not embedded in the program itself.

Transaction processing
A database management system must include concurrency control subsystems. This feature ensures that data remains consistent and valid during transaction processing even if several users update the same information.



Backup and recovery facilities

Recover from data loss caused by

- Incorrect transactions
- Hardware failure
- Software errors
- Data corruption
- User error
- Power interruption
- etc...

Backup and recovery facilities
Backup and recovery are methods that allow you to protect your data from loss. The database system provides a separate process, from that of a network backup, for backing up and recovering data. If a hard drive fails and the database stored on the hard drive is not accessible, the only way to recover the database is from a backup. If a computer system fails in the middle of a complex update process, the recovery subsystem is responsible for making sure that the database is restored to its original state. These are two more benefits of a database management system.

Data Modelling (A.2.8)

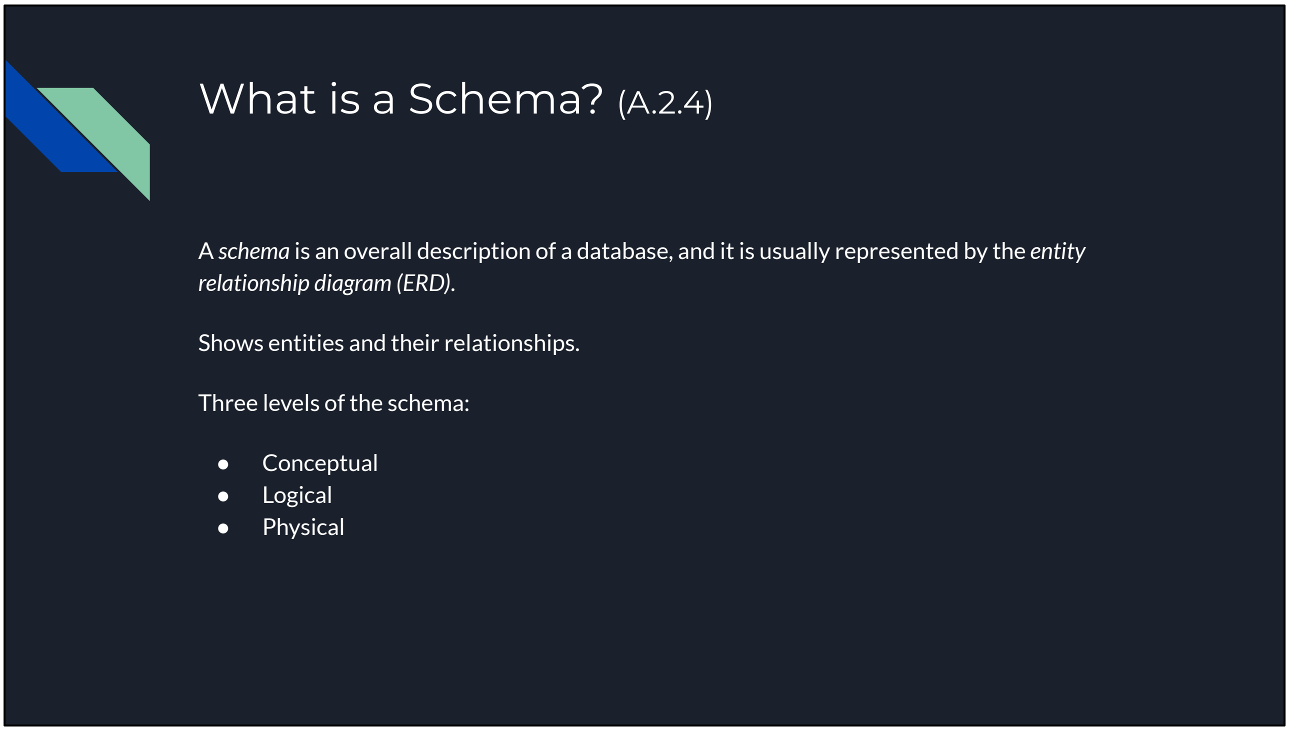
First step in the process of database design.

High-level, abstract design phase sometimes called Conceptual Design

The aim of the phase is to describe:

- The data contained in the database (e.g., entities: students, lecturers, courses, subjects)
- The relationships between data items (e.g., students are supervised by lecturers; lecturers teach courses)
- The constraints on data (e.g., student number has exactly eight digits; a subject has four or six units of credit only)

The *data model* is a collection of concepts or notations for describing data, data relationships, data semantics and data constraints. Most data models also include a set of basic operations for manipulating data in the database.



What is a Schema? (A.2.4)

A *schema* is an overall description of a database, and it is usually represented by the *entity relationship diagram (ERD).*

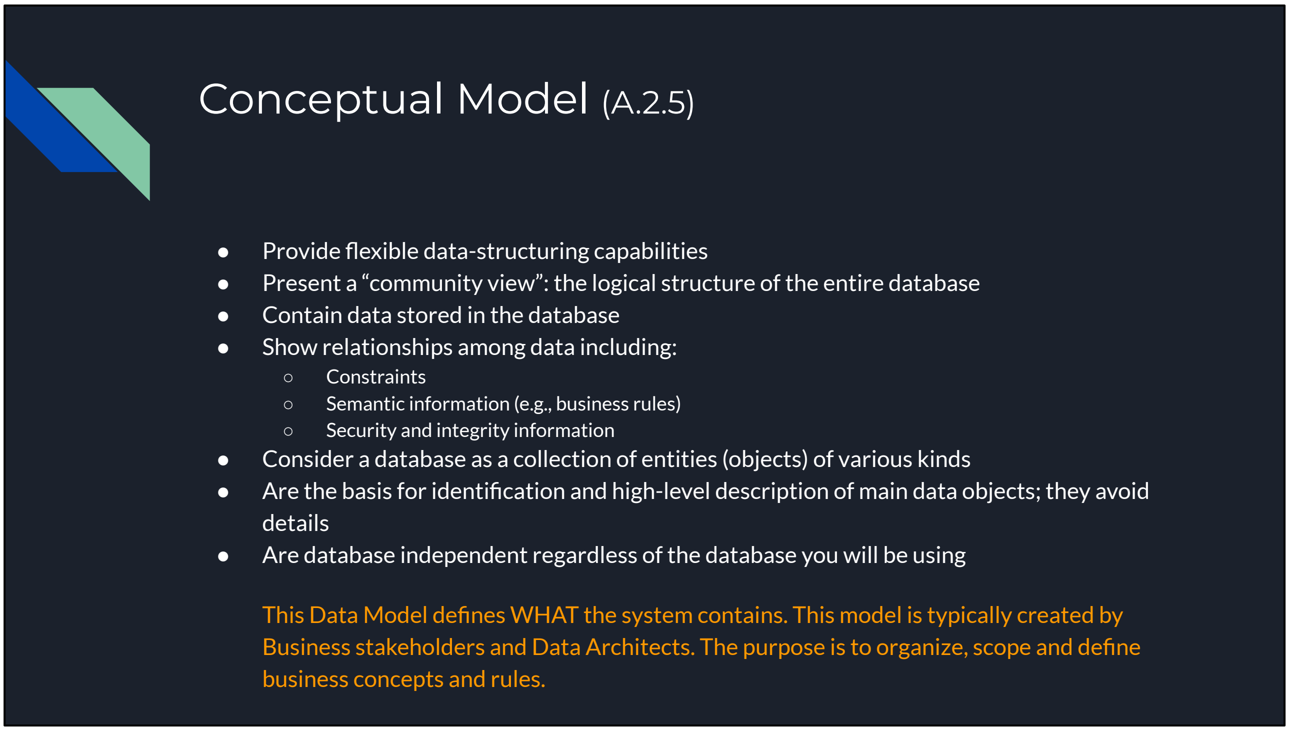
Shows entities and their relationships.

Three levels of the schema:

- Conceptual
- Logical
- Physical

Schemas
A *schema* is an overall description of a database, and it is usually represented by the *entity relationship diagram (ERD)*. There are many subschemas that represent external models and thus display external views of the data. Below is a list of items to consider during the design process of a database.

External schemas: there are multiple
Multiple subschemas: these display multiple external views of the data
Conceptual schema: there is only one. This schema includes data items, relationships and constraints, all represented in an ERD.
Physical schema: there is only one

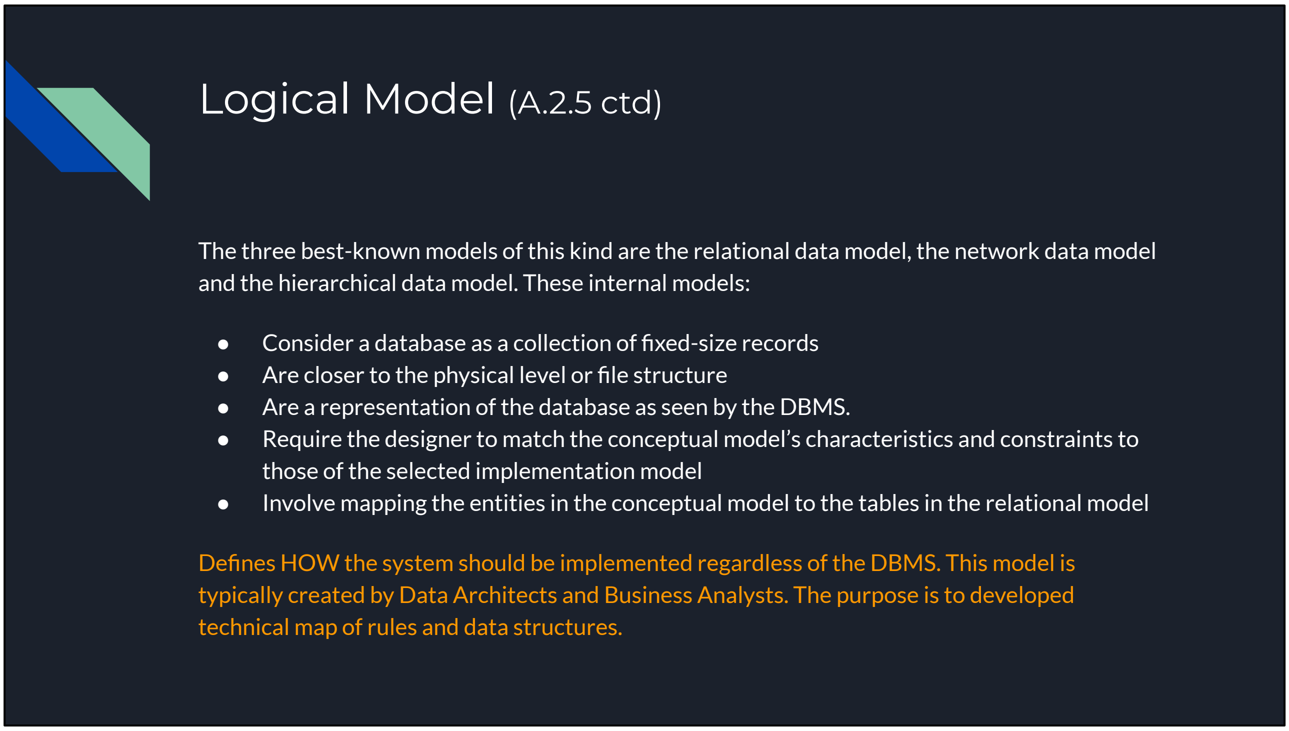


Conceptual Model (A.2.5)

- Provide flexible data-structuring capabilities
- Present a “community view”: the logical structure of the entire database
- Contain data stored in the database
- Show relationships among data including:
  - Constraints
  - Semantic information (e.g., business rules)
  - Security and integrity information
- Consider a database as a collection of entities (objects) of various kinds
- Are the basis for identification and high-level description of main data objects; they avoid details
- Are database independent regardless of the database you will be using

This Data Model defines WHAT the system contains. This model is typically created by Business stakeholders and Data Architects. The purpose is to organize, scope and define business concepts and rules.

High-level conceptual data models provide concepts for presenting data in ways that are close to the way people perceive data. A typical example is the entity relationship model, which uses main concepts like entities, attributes and relationships. An entity represents a real-world object such as an employee or a project. The entity has attributes that represent properties such as an employee’s name, address and birthdate. A relationship represents an association among entities; for example, an employee works on many projects. A relationship exists between the employee and each project.

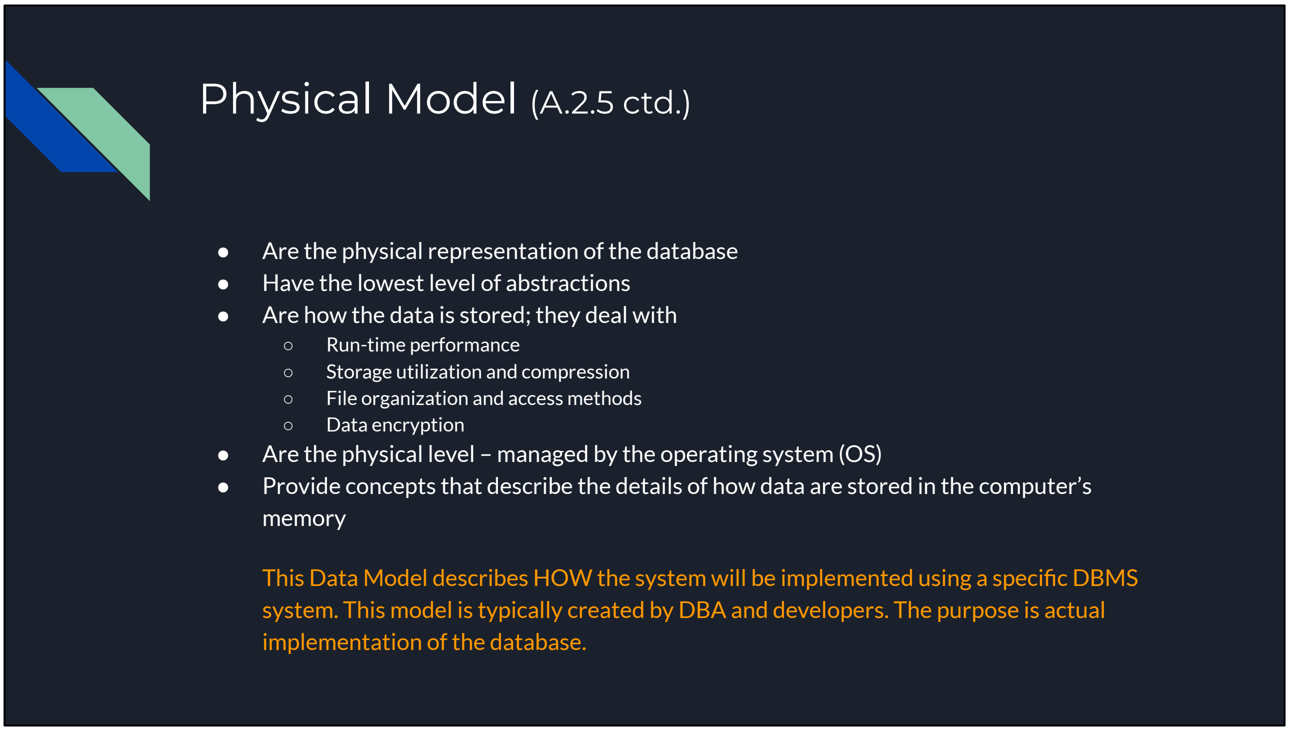


Logical Model (A.2.5 ctd)

The three best-known models of this kind are the relational data model, the network data model and the hierarchical data model. These internal models:

- Consider a database as a collection of fixed-size records
- Are closer to the physical level or file structure
- Are a representation of the database as seen by the DBMS.
- Require the designer to match the conceptual model's characteristics and constraints to those of the selected implementation model
- Involve mapping the entities in the conceptual model to the tables in the relational model

Defines HOW the system should be implemented regardless of the DBMS. This model is typically created by Data Architects and Business Analysts. The purpose is to developed technical map of rules and data structures.



Physical Model (A.2.5 ctd.)

- Are the physical representation of the database
- Have the lowest level of abstractions
- Are how the data is stored; they deal with
  - Run-time performance
  - Storage utilization and compression
  - File organization and access methods
  - Data encryption
- Are the physical level – managed by the operating system (OS)
- Provide concepts that describe the details of how data are stored in the computer's memory

This Data Model describes HOW the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.

https://www.visual-paradigm.com/support/documents/vpuserguide/3563/3564/85378_conceptual,l.html

http://uksanjay.blogspot.com/2012/06/difference-between-conceptual-logical.html

Data Dictionary (A.2.6)

Centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format. - *IBM Dictionary of Computing*

A collection of tables with metadata. - *Oracle*

A data dictionary contains metadata i.e data about the database. The data dictionary is very important as it contains information such as what is in the database, who is allowed to access it, where is the database physically stored etc. The users of the database normally don't interact with the data dictionary, it is only handled by the database administrators.

Data Dictionary ctd. (A.2.6 ctd.)

The data dictionary in general contains information about the following:

- Names of all the database tables and their schemas.
- Details about all the tables in the database, such as their owners, their security constraints, when they were created etc.
- Physical information about the tables such as where they are stored and how.
- Table constraints such as primary key attributes, foreign key information etc.
- Information about the database views that are visible.

Active and Passive Data Dictionaries

Active Data Dictionary - When a change is made to a database or the structure / schema of the database, the DBMS automatically updates it. Most databases use active data dictionaries.

Passive Data Dictionary - Data dictionary is maintained separate from the database. When a change is made to the structure / schema of the database, the data dictionary is not automatically updated and must be manually updated. Less useful and more rare than active data dictionaries.

Data Definition Language (DDL) (A.2.7)

A data definition or data description language (DDL) is a syntax similar to a computer programming language for defining data structures, especially database schemas.

DDL statements create, modify, and remove database objects such as tables, indexes, and users.

SQL (Structured Query Language) is one of the most popular DDLs.

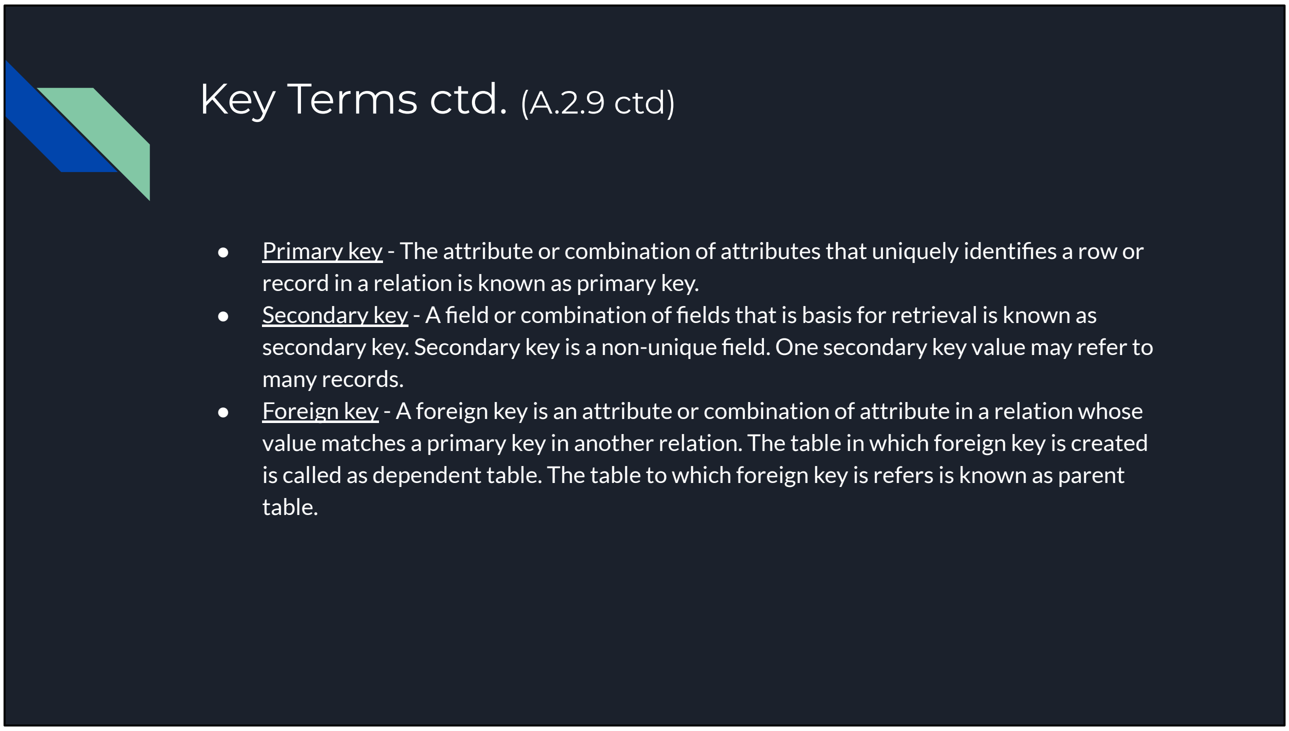
Examples:

- CREATE TABLE [*table name*] ([*column definitions*]) [*table properties*]
- DROP [*object type*] [*object name*]
- ALTER [*object type*] [*object name*] [*parameters*]

Other DDLs: JSON Schema, XML Schema

Key Terms (A.2.9)

- Table (*relation / file / class*) - A table is a collection of rows. A table usually has a name, although some tables are temporary and exist only to carry out a command. All the rows in a table have the same shape (in other words, every row in a table contains the same set of columns).
- Record (*tuple / row*) - A record is a collection of column values (fields). Every row in a table has the same shape (in other words, every row is composed of the same set of columns). If you are trying to model a real-world application, a row represents a real-world object.
- Field (*attribute / column*) - A field is the smallest unit of storage in a relational database. A column represents one piece of information about an object. Every column has a name and a data type. Columns are grouped into rows, and rows are grouped into tables.



Key Terms ctd. (A.2.9 ctd)

- Primary key - The attribute or combination of attributes that uniquely identifies a row or record in a relation is known as primary key.
- Secondary key - A field or combination of fields that is basis for retrieval is known as secondary key. Secondary key is a non-unique field. One secondary key value may refer to many records.
- Foreign key - A foreign key is an attribute or combination of attribute in a relation whose value matches a primary key in another relation. The table in which foreign key is created is called as dependent table. The table to which foreign key is refers is known as parent table.

A key is a single or combination of multiple fields. Its purpose is to access or retrieve data rows from table according to the requirement. The keys are defined in tables to access or sequence the stored data quickly and smoothly. They are also used to create links between different tables.

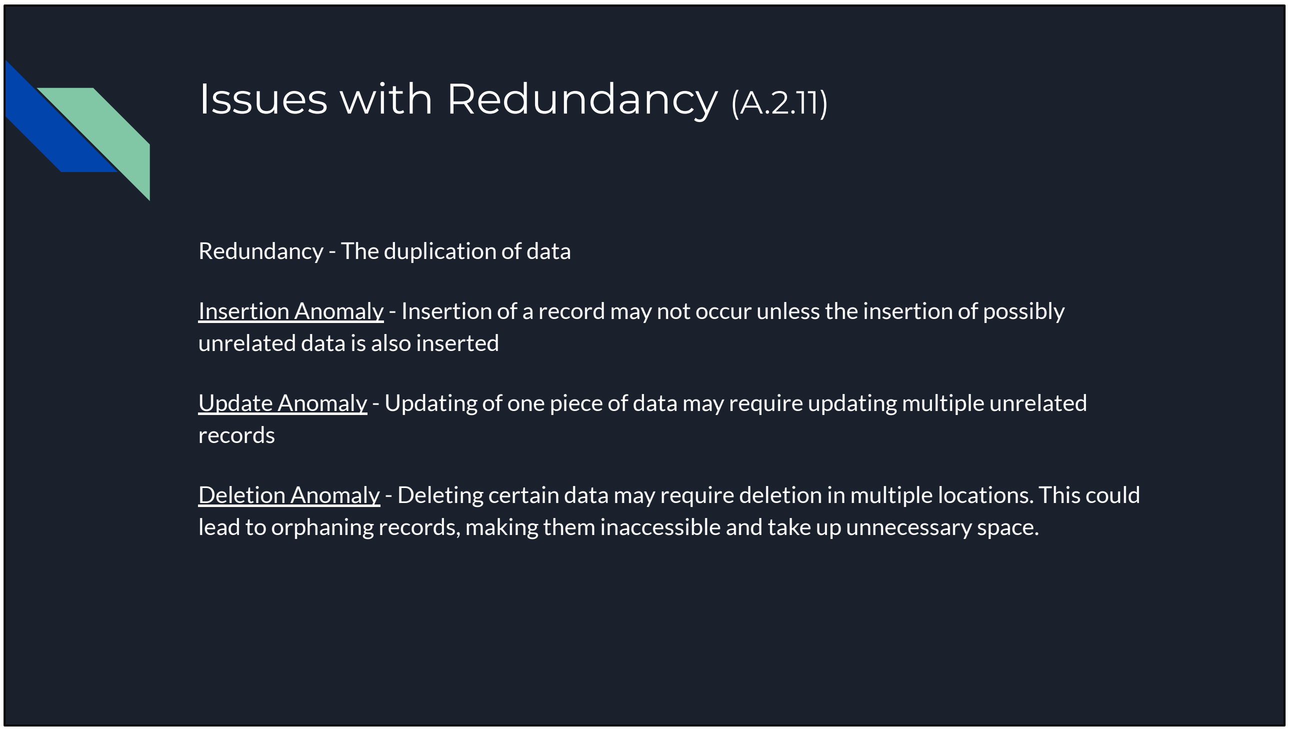
An attribute or combination of attributes that uniquely identify an entity/record in a relational table.

Key Terms ctd. (A.2.9 ctd)

- Candidate key - A relation can have only one primary key. It may contain many fields or combination of fields that can be used as primary key. One field or combination of fields is used as primary key. The fields or combination of fields that are not used as primary key are known as candidate key or alternate key.
- Composite primary key - A primary key that consists of two or more attributes is known as composite key.
- Join - A combination of two or more tables into a single view. Data from multiple tables is queried and compiled into a single view returned by the DBMS.

Table relationships (A.2.10)

- One-to-One
- One-to-Many
- Many-to-Many



Issues with Redundancy (A.2.11)

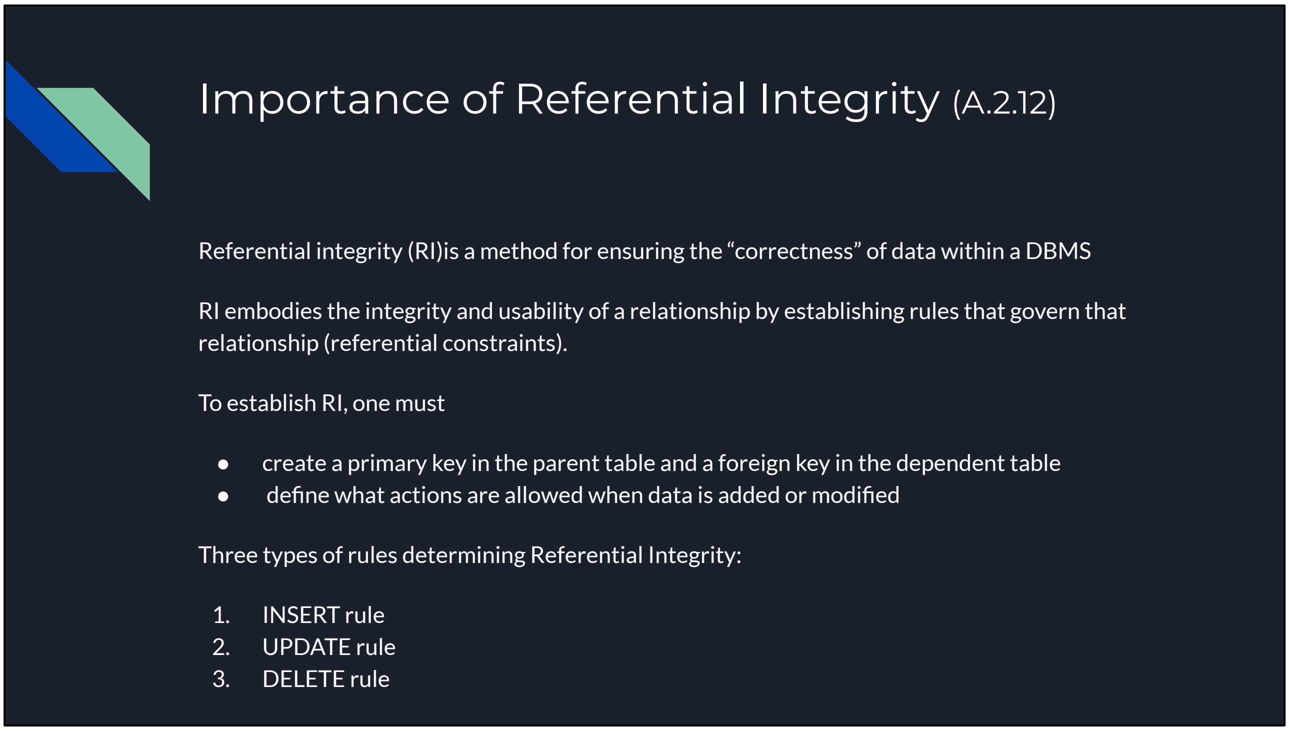
Redundancy - The duplication of data

Insertion Anomaly - Insertion of a record may not occur unless the insertion of possibly unrelated data is also inserted

Update Anomaly - Updating of one piece of data may require updating multiple unrelated records

Deletion Anomaly - Deleting certain data may require deletion in multiple locations. This could lead to orphaning records, making them inaccessible and take up unnecessary space.

https://www.geeksforgeeks.org/the-problem-of-redundancy-in-database/



Importance of Referential Integrity (A.2.12)

Referential integrity (RI)is a method for ensuring the “correctness” of data within a DBMS

RI embodies the integrity and usability of a relationship by establishing rules that govern that relationship (referential constraints).

To establish RI, one must

- create a primary key in the parent table and a foreign key in the dependent table
- define what actions are allowed when data is added or modified

Three types of rules determining Referential Integrity:

1. INSERT rule
2. UPDATE rule
3. DELETE rule

http://www.dbta.com/Columns/DBA-Corner/The-Importance-of-Referential-Constraints-for-Data-Integrity-119799.aspx

INSERT rule (IR A.2.12 ctd)

The INSERT rule indicates what will happen if you attempt to insert a value into a foreign key column without a corresponding primary key value in the parent table.

This should NEVER be allowed UNLESS the foreign key is being set to null.

UPDATE rule (IR A.2.12 ctd)

The UPDATE rule controls data modification such that a foreign key value cannot be updated to a value that does not correspond to a primary key value in the parent table.

Do we allow the primary key to be updated? If it is, what happens to all the foreign key references to the previous primary key?

Do we update all previous references? Set previous references to null? How do we find previous references?

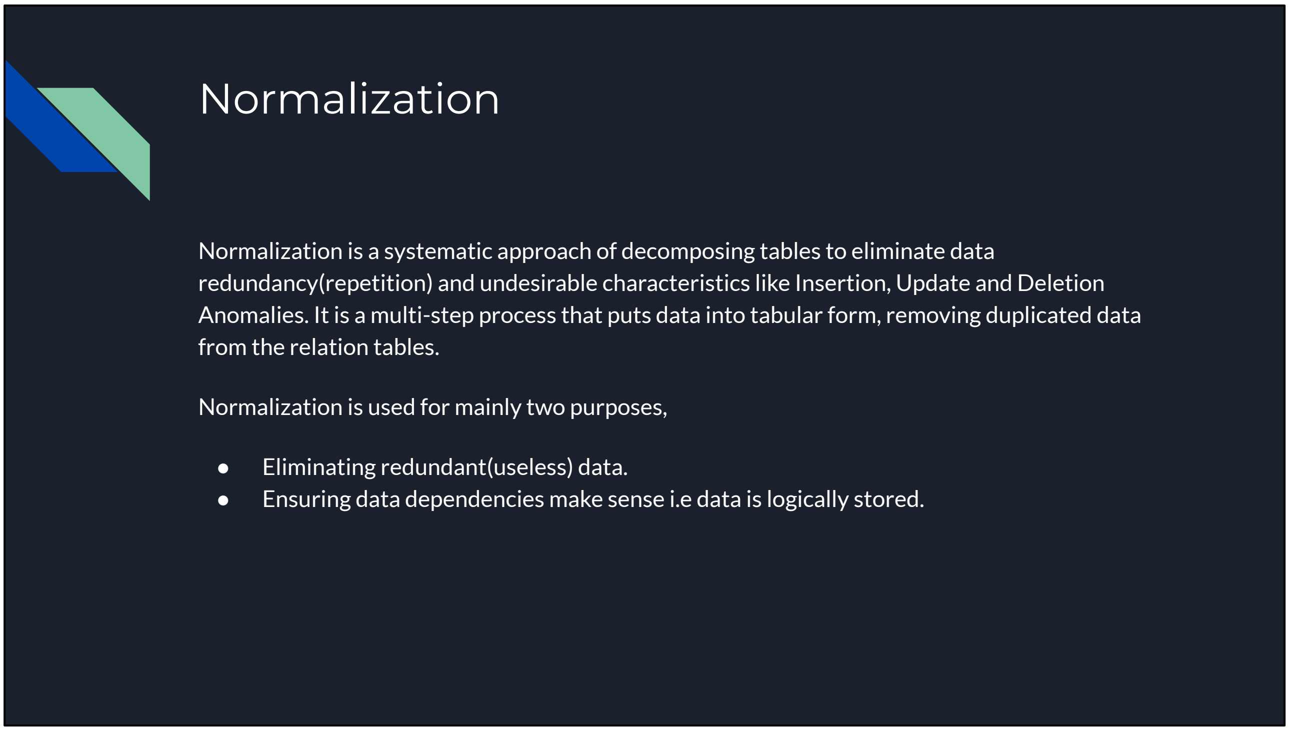
DELETE rule (IR A.2.12 ctd)

DELETE rules define what happens when an attempt is made to delete a row from the parent table.

Should we allow a record to be deleted while there are other records referencing this one?

What do we do with those references?

- Delete records?
- Update foreign key to null?
- Don’t allow deletion?



Normalization

Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

- Eliminating redundant(useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

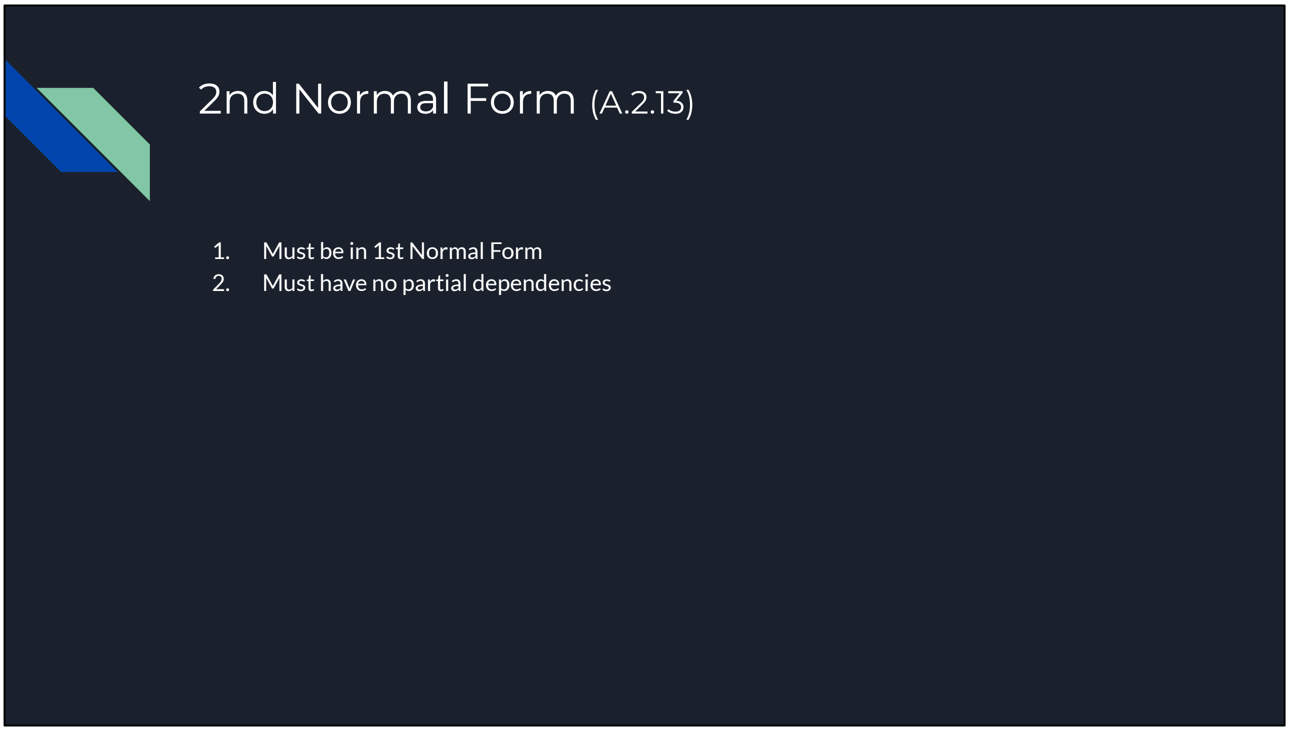
https://studytonight.com/dbms/database-normalization.php (Video)

1st Normal Form (A.2.13)

Each field contains ONE and only ONE value

Course	Student
Computer Science SL	Harrison, David, Paul

Course	Student
Computer Science SL	Harrison
Computer Science SL	David
Computer Science SL	Paul



2nd Normal Form (A.2.13)

1. Must be in 1st Normal Form
2. Must have no partial dependencies

https://en.wikipedia.org/wiki/Database_normalization

3rd Normal Form (A.2.13)

1. Must be in 2nd Normal Form
2. Must not have any transitive dependencies

Activities

- A.2.14 - Describe the characteristics of a normalized database.
- A.2.15 - Evaluate the appropriateness of the different data types.
- A.2.16 - Construct an Entity-Relationship Diagram for a given scenario.
- A.2.17 - Construct a relational database to 3NF using object such as tables, queries, forms, reports, and macros.

Queries provide views (A.2.18)

A view is a particular set of data stored in the database and returned by a query. Views can contain data from multiple tables combined with a JOIN.

Different queries can present data in different ways, even when the data is the same data.

SELECT (given_name, family_name, age) FROM students SORT BY (age)

SELECT (given_name, family_name, age) FROM students SORT BY (given_name)

Same data, different presentation

Simple vs Complex query (A.2.19)

Simple queries select basic data.

Complex queries can select data based on some logic using AND, OR, NOT.

Complex queries can often summarize data by summation, averaging, etc…

Complex queries can create fields that are not part of any table based on the data held in other tables (like sums or averages)