

# DB Cheatsheet

## Ch1. An Overview of Database Systems

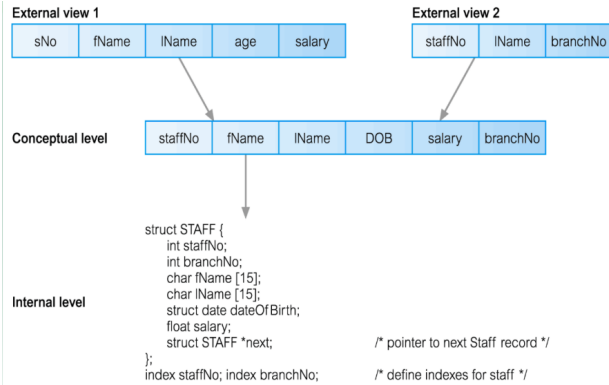
### 1. Databases and File Processing Systems

File Processing Systems	Database Systems
Data are stored separately for each application	Data are organized centrally in a database
Uncontrolled data redundancy	Controlled data redundancy
Programs are tightly coupled with data structures	Program-data independence is ensured
Difficult to maintain and extend when requirements change	Easy to maintain and extend
Data sharing among applications is difficult	Data sharing is supported for multiple users and applications

### 2. Three-Schema Architecture and Data Independence

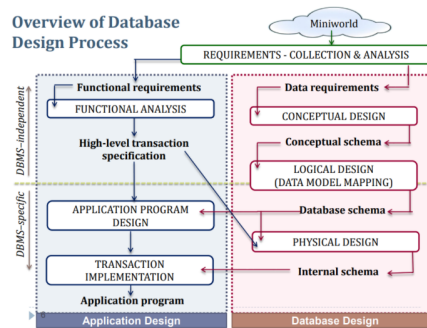
A database system adopts the three-schema architecture:

- **External level:** user views tailored to different user groups.
- **Conceptual level:** a complete logical description of the entire database.
- **Internal level:** the physical storage structure of the database.



## Ch2. ERD

### 1. Overview of DB design process



### 2. ERD

Illustration	Attribute Type	Description
	Simple Attribute	Has a single, atomic value that cannot be further divided.
	Composite Attribute	Consists of multiple components that can be meaningfully separated.
	Multi-valued Attribute	May have multiple values for a single entity. May have constrain the number of values allowed for each individual identity.
	Derived Attribute	Its value is derived from other related attributes rather than stored directly.
	Complex Attribute	A combination of composite and multi-valued attributes.
	Key Attribute	An attribute (or set of attributes) whose values uniquely identify each entity in an entity set.

**Degree** of a relationship type: Number of participating entity types. (recursive is unary)

**Cardinality ratios:** the maximum number of instances an entity can participate in (1:1, 1:n, n:1, n:m).

**Participation:** the minimum number of relationship instances that each entity can participate in. (Total participation: every entity participates in, Partial: some (not every) entities).

**Attribute of a relationship type:** Relationship types can also have attributes.

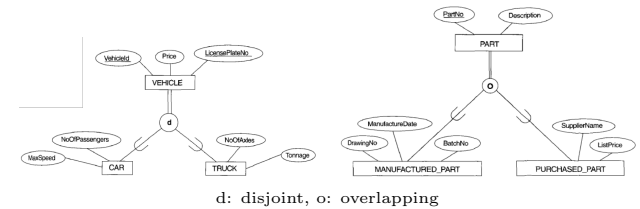
## Ch2. ERD

### Weak Entities Type

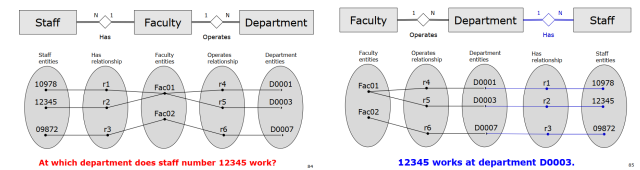
- A weak entity type does not have its own key attribute.
- Weak entities are identified by an owner (identifying) entity type, and an identifying relationship, together with a partial key.
- A weak entity type has **total participation** (existence dependency) in its identifying relationship.
- A **partial key** uniquely identifies weak entities related to the same owner entity.

### 3. EERD

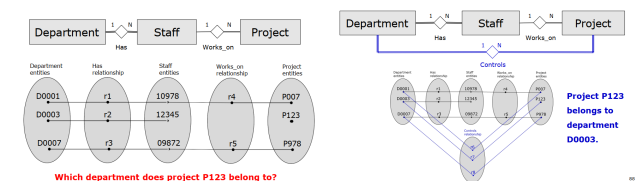
- **Specialization:** process of defining subclasses of an entity type with
  - specific attributes,
  - specific relationship types.
- **Generalization:** reverse abstraction process that
  - identifies common features of multiple entity types,
  - combines them into a single superclass.



### Fan trap



### Chamsp trap



## 1. Data Model

A **data model** consists of three components: **data structures**, **operations**, and **integrity constraints**.

### Data Structures

In the relational data model, data is represented using **relations**.

An  **$n$ -ary relation**  $R$  defined on domains  $S_1, S_2, \dots, S_n$  is a subset of their Cartesian product:

$$R \subseteq S_1 \times S_2 \times \dots \times S_n$$

Each row of  $R$  is an  $n$ -**tuple**:

$$t = \langle v_1, v_2, \dots, v_n \rangle, \quad v_i \in S_i$$

**Inherent (implicit) properties:**

- Rows represent  $n$ -tuples; ordering of rows is immaterial.
- All rows are distinct.
- Column ordering is significant and corresponds to  $S_1, S_2, \dots, S_n$ .
- Each column is labeled by an **attribute** indicating its domain.

**Degree and Cardinality:**

- **Degree:** number of attributes ( $n$ ).
- **Cardinality:** number of tuples ( $|R|$ ).

### Operations

Operations specify how data can be retrieved and manipulated. Core relational algebra operators:

- **Selection** ( $\sigma$ ), **Projection** ( $\pi$ ), **Union** ( $\cup$ ), **Set Difference** ( $-$ ), **Cartesian Product** ( $\times$ ), **Join** ( $\bowtie$ ).

Relational operators satisfy the **closure property**.

### Integrity Constraints

Integrity constraints define valid database states:

- **Domain constraint:** attribute values must belong to the domain (e.g.,  $\text{Age} \geq 0$ ).
- **Key constraint:** a key is a **minimal superkey**; key values must be unique (e.g.,  $\text{StudentID}$  identifies  $\text{STUDENT}$ ).
- **Constraints on nulls:** specify whether NULL values are permitted for attributes.
- **Entity integrity constraint:** primary key attributes cannot have NULL values.
- **Referential integrity constraint:** foreign key values must reference a primary key value or be NULL (e.g.,  $\text{STUDENT.DeptID} \rightarrow \text{DEPARTMENT.DeptID}$ ).

## 2. Data model mapping

1. **Map Regular (Strong) Entity Types:** Create a relation for each strong entity type; include all simple attributes. Primary key = entity key.
2. **Map Weak Entity Types:** Create a relation for the weak entity including its simple attributes. Primary key = (partial key + primary key of owner). Owner key is a foreign key.
3. **Map Binary 1:1 Relationships:** Choose one relation (prefer total participation) and include the primary key of the other as a foreign key; add relationship attributes.
4. **Map Binary 1:N Relationships:** Include the primary key of the 1-side as a foreign key in the N-side relation; add relationship attributes to the N-side.
5. **Map Binary M:N Relationships:** Create a new relation whose primary key is the combination of the primary keys of participating entities; include relationship attributes.
6. **Map Multivalued Attributes:** Create a new relation with attributes (attribute value + owner primary key). Primary key = both attributes.
7. **Map  $n$ -ary Relationships ( $n > 2$ ):** Create a new relation including primary keys of all participating entities as foreign keys; primary key is their combination.
8. **Map Specialization / Generalization:** Use one of the following:
  - Superclass + subclass relations
  - Single relation with type attribute
  - Multiple relations for subclasses only
9. **Map Categories (Union Types):** Create a relation with a surrogate key and foreign keys referencing each superclass; enforce membership constraint.

## 3. Relational Algebra

Operation	Purpose	Notation
SELECT	Selects all tuples that satisfy the selection condition from a relation $R$ .	
PROJECT	Produces a new relation with only some of the attributes of $R$ , and removes duplicate tuples.	
THETA JOIN	Produces all combinations of tuples from $R_1$ and $R_2$ that satisfy the join condition.	
EQUIJOIN	Produces all combinations of tuples from $R_1$ and $R_2$ that satisfy a join condition with only equality comparisons.	
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of $R_2$ are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	
UNION	Produces a relation that includes all the tuples in $R_1$ or $R_2$ or both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	
INTERSECTION	Produces a relation that includes all the tuples in both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	
DIFFERENCE	Produces a relation that includes all the tuples in $R_1$ that are not in $R_2$ ; $R_1$ and $R_2$ must be union compatible.	
CARTESIAN PRODUCT	Produces a relation that has the attributes of $R_1$ and $R_2$ and includes as tuples all possible combinations of tuples from $R_1$ and $R_2$ .	
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in $R_1$ in combination with every tuple from $R_2(Y)$ , where $Z = X \cup Y$ .	