DB Cheatsheet

Ch1. An Overview of Database Systems

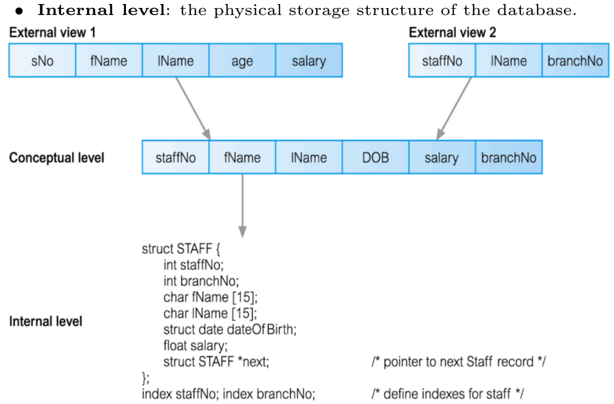
1. Databases and File Processing Systems

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

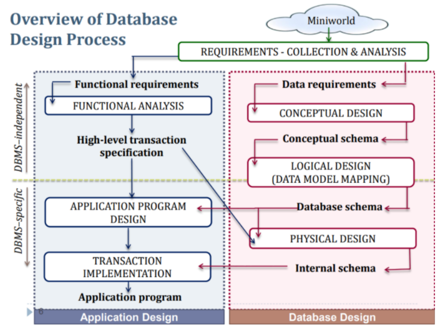
A database system adopts the three-schema architecture:

- **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.

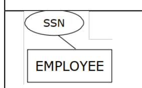
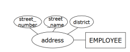
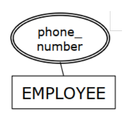
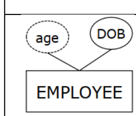
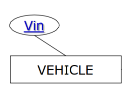


Ch2.ERD

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.

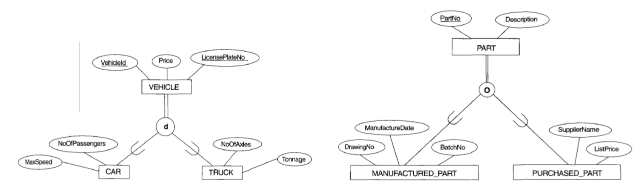
Ch2. ERD

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.

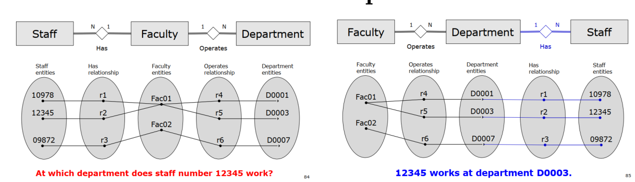
3. EERD

- **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.

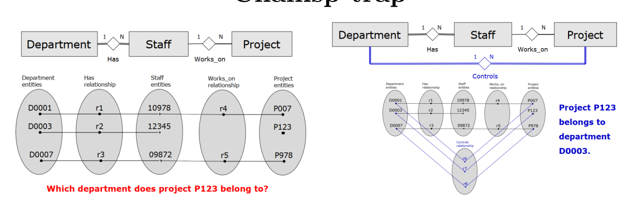


d: disjoint, o: overlapping

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 $S_1, S_2, \ldots, S_n$ is a subset of their Cartesian product:

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

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

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

Inherent (implicit) properties:

- Rows represent n-tuples; ordering of rows is immaterial.
- All rows are distinct.
- Column ordering is significant and corresponds to $S_1, S_2, \ldots, S_n$.
- 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** (σ), **Projection** (π), **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., Age ≥ 0).
- **Key constraint**: a key is a **minimal superkey**; key values must be unique (e.g., StudentID identifies 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., STUDENT.DeptID $\rightarrow$ 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 $Z = X \cup Y$.	

SELECT: $\sigma_{<conditions>}(R)$

- Retrieve the tuples in relation R that satisfy *<conditions>*.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

Retrieve all the information of each employee who is with department 4 and salary > 25000 or with department 5 and salary > 30000.

$\sigma_{(Dno=4 \text{ AND } Salary>25000) \text{ OR } (Dno=5 \text{ AND } Salary>30000)}$**(EMPLOYEE)**

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

88

SELECT: $\sigma_{<conditions>}(R)$

- Retrieve the tuples in relation R that satisfy *<conditions>*.
- The relation S = $\sigma_{<conditions>}(R)$ has the same schema (same attributes) as R.
- A cascade (sequence) of SELECT operations in any order:
  - $\sigma_{<cond1>}(\sigma_{<cond2>}(R)) = \sigma_{<cond1> \text{ AND } <cond2>}(R)$
- SELECT is commutative:
  - $\sigma_{<condition1>}(\sigma_{<condition2>}(R)) = \sigma_{<condition2>}(\sigma_{<condition1>}(R))$
- The number of tuples in the result of SELECT is less than or equal to the number of tuples in the input relation R

Retrieve all the information of each employee who is with department 4 and salary > 25000.

$\sigma_{Dno=4 \text{ AND } Salary>25000}$**(EMPLOYEE)**
= $\sigma_{Dno=4}(\sigma_{Salary>25000}$**(EMPLOYEE))**
= $\sigma_{Salary>25000}(\sigma_{Dno=4}$ **(EMPLOYEE))**

89

PROJECT: $\pi_{<Attribute\ list>}(R)$

- Return a relation of distinct tuples from relation R with the attributes listed in *<Attribute list>*

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07	[illegible]	[illegible]	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03	[illegible]	[illegible]	25000	987654321	4
James	E	Borg	888665555	1937-11	[illegible]	[illegible]	55000	NULL	1

Retrieve Ssn, date of birth, and department number of each employee.

$\pi_{Ssn, Bdate, Dno}$ **(EMPLOYEE)**

Ssn	Bdate	Dno
123456789	1965-01-09	5
333445555	1955-12-08	5
999887777	1968-01-19	4
987654321	1941-06-20	4
666884444	1962-09-15	5
453453453	1972-07-31	5
987987987	1969-03-29	4
888665555	1937-11-10	1

π_{Dno} **(EMPLOYEE)**

Dno
5
4
1

90

CARTESIAN PRODUCT (CROSS JOIN): R x S

- Produce a new relation *T* by combining every member (tuple) from one relation (set) with every member (tuple) from the other relation (set)
  - Degree of *T* = Degree of R + Degree of S
  - Cardinality of *T* = |T| = |R|*|S|

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Return all the combinations of departments and their locations.
DEPARTMENT x DEPT_LOCATIONS?

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

DEPARTMENT x DEPT_LOCATIONS

Dname	Dnumber	Mgr_ssn	Mgr_start_date	Dnumber	Dlocation
Research	5	333445555	1988-05-22	1	Houston
Research	5	333445555	1988-05-22	4	Stafford
Research	5	333445555	1988-05-22	5	Bellaire
Research	5	333445555	1988-05-22	5	Sugarland
Research	5	333445555	1988-05-22	5	Houston
Administration	4	987654321	1995-01-01	1	Houston
Administration	4	987654321	1995-01-01	4	Stafford
Administration	4	987654321	1995-01-01	5	Bellaire
Administration	4	987654321	1995-01-01	5	Sugarland
Administration	4	987654321	1995-01-01	5	Houston
Headquarters	1	888665555	1981-06-19	1	Houston
Headquarters	1	888665555	1981-06-19	4	Stafford
Headquarters	1	888665555	1981-06-19	5	Bellaire
Headquarters	1	888665555	1981-06-19	5	Sugarland
Headquarters	1	888665555	1981-06-19	5	Houston

92

THETA JOIN: $R \bowtie_{<conditions>} S$

- Produce a new relation Q with *n + m* attributes Q (A_1, A_2, … , A_n, B_1, B_2, … , B_m) in that order; Q has one tuple for each combination of tuples—one from R (A_1, A_2, … , A_n) and one from S (B_1, B_2, … , B_m)—whenever the combination satisfies the join condition.
- $R \bowtie_{<conditions>} S = \sigma_{<conditions>}(R \times S)$

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Return all the combinations of Research department with Houston location.
DEPARTMENT $\bowtie_{Dname = \text{'Research'} \text{ AND } Dlocation = \text{'Houston'}}$ **DEPT_LOCATIONS?**

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Return all the combinations of Research department with Houston location.
DEPARTMENT $\bowtie_{Dname = \text{'Research'} \text{ AND } Dlocation = \text{'Houston'}}$ **DEPT_LOCATIONS**

EQUIJOIN: THETA JOIN with *equality* (=) comparisons only.

- $R \bowtie_{A=B} S = \sigma_{A=B}(R \times S)$, where A are attributes in R and B are attributes in S.

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Retrieve all the information of each department and its locations.
DEPARTMENT $\bowtie_{Dnumber = Dnumber}$ **DEPT_LOCATIONS?**

96

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Retrieve all the information of each department and its locations.
DEPARTMENT $\bowtie_{Dnumber = Dnumber}$ **DEPT_LOCATIONS**

Dname	Dnumber	Mgr_ssn	Mgr_start_date	Dnumber	Dlocation
Research	5	333445555	1988-05-22	5	Bellaire
Research	5	333445555	1988-05-22	5	Sugarland
Research	5	333445555	1988-05-22	5	Houston
Administration	4	987654321	1995-01-01	4	Stafford
Headquarters	1	888665555	1981-06-19	1	Houston

98

NATURAL JOIN: EQUIJOIN by *getting rid of the second attribute* in an equality condition when the two join attributes (or each pair of join attributes) have the *same name* in both relations

- R * S

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Retrieve all the information of each department and its locations.
DEPARTMENT * DEPT_LOCATIONS?

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Retrieve all the information of each department and its locations.
DEPARTMENT $\bowtie_{\text{Dnumber = Dnumber}}$ DEPT_LOCATIONS

Dname	Dnumber	Mgr_ssn	Mgr_start_date	Dnumber	Dlocation
Research	5	333445555	1988-05-22	5	Bellaire
Research	5	333445555	1988-05-22	5	Sugarland
Research	5	333445555	1988-05-22	5	Houston
Administration	4	987654321	1995-01-01	4	Stafford
Headquarters	1	888665555	1981-06-19	1	Houston

Retrieve all the information of each department and its locations.
DEPARTMENT * DEPT_LOCATIONS

Dname	Dnumber	Mgr_ssn	Mgr_start_date	Dlocation
Research	5	333445555	1988-05-22	Bellaire
Research	5	333445555	1988-05-22	Sugarland
Research	5	333445555	1988-05-22	Houston
Administration	4	987654321	1995-01-01	Stafford
Headquarters	1	888665555	1981-06-19	Houston

From *equijoin* to *natural join*

OUTER JOIN:

- A set of operations, called *outer joins*, can be used when we want to keep all the tuples in R, or all those in S, or all those in both relations in the result of the JOIN, regardless of whether or not they have matching tuples in the other relation.
  - Left outer join, right outer join, full outer join
- The join operations where only matching tuples are kept in the result are called *inner joins*.
  - Theta join, equijoin, natural join

LEFT OUTER JOIN: R ⟕ S

- In addition to the result of the corresponding inner join, the LEFT OUTER JOIN operation keeps every tuple in the *first*, or *left*, *relation R* in R ⟕ S; if *no* matching tuple is found in S, then the attributes of S in the join result are filled or "padded" with null values.

Return the information of all the employees and their departments that they manage.
EMPLOYEE $\bowtie_{\text{Ssn = Mgr\_ssn}}$ DEPARTMENT

Return the information of all the employees and their departments that they manage if any.
EMPLOYEE ⟕$_{\text{Ssn = Mgr\_ssn}}$ DEPARTMENT

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

Return the information of all the employees and their departments that they manage.
EMPLOYEE $\bowtie_{\text{Ssn = Mgr\_ssn}}$ DEPARTMENT

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno	Dname	Dnumber	Mgr_ssn	Mgr_start_date
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	Research	5	333445555	1988-05-22
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4	Administration	4	987654321	1995-01-01
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1	Headquaters	1	888665555	1981-06-09

Return the information of all the employees and their departments that they manage if any.
EMPLOYEE ⟕$_{\text{Ssn = Mgr\_ssn}}$ DEPARTMENT

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno	Dname	Dnumber	Mgr_ssn	Mgr_start_date
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	Research	5	333445555	1988-05-22
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4	Administration	4	987654321	1995-01-01
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1	Headquaters	1	888665555	1981-06-09
John	B	Smith	123456789	1965-01-09	732 Fondren, Houston, TX	M	30000	333445555	5	NULL	NULL	NULL	NULL
Alicia	J	Zelaya	999778888	...	...	...	...	...	4	NULL	NULL	NULL	NULL
Ramesh	K	Narayan	...	...	...	...	...	...	5	NULL	NULL	NULL	NULL
Joyce	A	English	...	...	...	...	...	...	5	NULL	NULL	NULL	NULL
Ahmad	V	Jabbar	...	...	...	...	...	...	4	NULL	NULL	NULL	NULL

RIGHT OUTER JOIN: R ⟖ S

- In addition to the result of the corresponding inner join, the RIGHT OUTER JOIN operation keeps every tuple in the *second*, or *right*, *relation S* in R ⟖ S; if *no* matching tuple is found in R, then the attributes of R in the join result are filled or "padded" with null values.

FULL OUTER JOIN: R ⟗ S

- In addition to the result of the corresponding inner join, the FULL OUTER JOIN operation keeps every tuple in both the *left* and the *right relations* when *no* matching tuples are found, padding them with null values as needed.

SEMI-JOIN: T1 $\ltimes_{T1.X=T2.Y}$ T2

- A tuple of T1 is returned as soon as T1.X finds a match with any value of T2.Y *without* searching for *further* matches.
  - This is in contrast to finding all possible matches in inner join.
- Semi-join is generally used for unnesting EXISTS, IN, and ANY subqueries.
- SEMI-JOIN can be extended with any *conditions*.

Return the information of all the departments *with at least one* employee who has salary > 30000.

DEPARTMENT $\ltimes_{\text{Dnumber=Dno}}$ ($\sigma_{\text{Salary>30000}}$(EMPLOYEE))

ANTI JOIN: T1 $\overline{\ltimes}_{X = Y}$ T2

- A tuple of T1 is returned, only if T1.X *does not* match with any value of T2.Y. A tuple of T1 is rejected as soon as T1.X finds a match with any value of T2.Y.
- Anti-join is used for unnesting NOT EXISTS, NOT IN, and ALL subqueries.
- ANTI JOIN can be extended with any *conditions*.

Return the information of all the employees who *do not* work in any department which was managed since 1990.

EMPLOYEE $\overline{\ltimes}_{\text{Dno=Dnumber}}$ ($\sigma_{\text{Mgr\_start\_date>='1990-01-01'}}$(DEPARTMENT))

UNION: R U S

- *Union compatibility*: R and S has the same degree and each corresponding pair of attributes has the same domain.
- Produce a new relation that includes all tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.

RESULT1 U RESULT2

RESULT1	SSN
	123456789
	333445555
	666884444
	453453453

RESULT2	SSN
	333445555
	888665555

RESULT	SSN
	123456789
	333445555
	666884444
	453453453
	888665555

INTERSECTION: R ∩ S

- *Union compatibility*: R and S has the same degree and each corresponding pair of attributes has the same domain.
- Produce a new relation that includes all tuples that are in both R and S.

STUDENT	FN	LN
	Susan	Yao
	Ramesh	Shah
	Johnny	Kohler
	Barbara	Jones
	Amy	Ford
	Jimmy	Wang
	Ernest	Gilbert

INSTRUCTOR	FNAME	LNAME
	John	Smith
	Ricardo	Browne
	Susan	Yao
	Francis	Johnson
	Ramesh	Shah

STUDENT ∩ INSTRUCTOR

DIFFERENCE: R – S

- *Union compatibility*: R and S has the same degree and each corresponding pair of attributes has the same domain.
- Produce a new relation that includes all tuples that are in R but not in S.

STUDENT	FN	LN
	Susan	Yao
	Ramesh	Shah
	Johnny	Kohler
	Barbara	Jones
	Amy	Ford
	Jimmy	Wang
	Ernest	Gilbert

INSTRUCTOR	FNAME	LNAME
	John	Smith
	Ricardo	Browne
	Susan	Yao
	Francis	Johnson
	Ramesh	Shah

STUDENT - INSTRUCTOR

FN	LN
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR - STUDENT

FNAME	LNAME
John	Smith
Ricardo	Browne
Francis	Johnson

DIVISION: T(Y) = R(Z) ÷ S(X), where X ⊆ Z, Y = Z - X

- Produce a new relation T each tuple of which appears in R in combination with *every* tuple in S
- A relation T(Y) includes a tuple *t* if tuples t_R appear in R with t_R[Yl = t, and with $t_R[X] = t_S$ for every tuple t_S in S.
- DIVISION can be expressed for the *all* condition as a sequence of π, x, and – operations.

Return the employees who worked on *all* the projects that Smith worked:
SSN_PNOS ÷ SMITH_PNOS

The projects of all employees

SSN_PNOS	ESSN	PNO
	123456789	1
	123456789	2
	666884444	3
	453453453	1
	453453453	2
	333445555	2
	333445555	3
	333445555	10
	333445555	20
	999887777	30
	999887777	10
	987987987	10
	987987987	30
	987654321	30
	987654321	20
	888665555	20

Smith's projects

SMITH_PNOS	PNO
	1
	2

AGGREGATION: *<a grouping attribute list>* $\Im$ *<function list>* (R)

- <grouping attributes> is a list of attributes of R.
- <function list> is a list of (<function> <attribute>) pairs. In each such pair, <function> is one of the allowed *aggregate* functions—such as SUM, AVERAGE, MAXIMUM, MINIMUM, COUNT—and <attribute> is an attribute of R.
- The resulting relation has the grouping attributes plus one attribute for each element in the function list.

How many employees work in the company?
How many employees work in each department of the company?

AGGREGATION: *<a grouping attribute list>* $\Im$ *<function list>* (R)

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

How many employees work in the company?

$\Im$ COUNT Ssn **(EMPLOYEE)**

How many employees work in each department of the company?

Dno $\Im$ COUNT Ssn **(EMPLOYEE)**

COUNT_Ssn
8

Dno	COUNT_Ssn
5	4
4	3
1	1

1. SQL Language

- **Data Definition Language (DDL)**: used to define and modify database schema (`CREATE`, `ALTER`, `DROP`)
- **Data Manipulation Language (DML)**: used to retrieve and manipulate data (`SELECT`, `INSERT`, `DELETE`, `UPDATE`)
- **Data Control Language (DCL)**:
  - *Access control*: `GRANT`, `REVOKE`
  - *Transaction control*: `COMMIT`, `ROLLBACK`, `SET AUTOCOMMIT OFF`

Note: SQL is **case-insensitive** and is often **extended by DBMSs** with additional proprietary features.

2. DDL Language

CREATE SCHEMA is used to define a logical container (namespace) that groups database objects such as tables, views, domains, and constraints. It helps organize the database, avoid name conflicts, and support authorization.

Syntax: `CREATE SCHEMA schema_name [AUTHORIZATION user_name];`

CREATE TABLE defines the structure of a relation, including attributes, data types, and integrity constraints.

Syntax:

```
CREATE TABLE table_name (
attribute_name data_type [attribute_constraint],
...
[table_constraint]
);
```

CREATE DOMAIN is used to define a user-defined data type with optional constraints that can be reused across multiple tables. It improves consistency and reduces redundancy in schema definitions.

Syntax:

```
CREATE DOMAIN domain_name AS data_type
[DEFAULT default_value]
[CHECK (VALUE condition)];
```

- Domain constraints are enforced wherever the domain is used.
- `VALUE` refers to the attribute value being checked.
- Changing a domain affects all attributes defined using that domain.

Attribute-level constraints:

- `NOT NULL`: disallows NULL values for the attribute.
- `DEFAULT value`: assigns a default value when none is provided.
- `UNIQUE`: ensures all values in the attribute are distinct.
- `CHECK(condition)`: restricts attribute values to satisfy a condition.
- `PRIMARY KEY`: uniquely identifies each tuple and disallows NULLs.
- `REFERENCES table(attribute)`: enforces referential integrity with a referenced table.

Table-level constraints:

- `PRIMARY KEY (A, B)`: defines a composite primary key over multiple attributes.
- `UNIQUE (A, B)`: enforces uniqueness over a combination of attributes.
- `FOREIGN KEY (A) REFERENCES R(B)`: specifies a foreign key relationship between tables.
- `CHECK(attr_name condition)`: restricts tuples based on a condition involving multiple attributes.

Referential actions:

- `ON DELETE CASCADE`: deletes referencing tuples when the referenced tuple is deleted.
- `ON DELETE SET NULL`: sets foreign key values to NULL when the referenced tuple is deleted.
- `ON UPDATE CASCADE`: updates foreign key values when the referenced key is updated.
- `RESTRICT`: prevents deletion or update of a referenced tuple (default behavior).

ALTER TABLE is used to modify the structure of an existing table, including attributes, constraints, and referential actions, without recreating the table.

Common operations:

Operation	Syntax
Add attribute	`ALTER TABLE table_name ADD attribute data_type [constraint];`
Drop attribute	`ALTER TABLE table_name DROP attribute RESTRICT (default)/CASCADE;`
Modify attribute	`ALTER TABLE table_name ALTER attribute SET data_type;`
Add constraint	`ALTER TABLE table_name ADD CONSTRAINT cname constraint_definition;`
Drop constraint	`ALTER TABLE table_name DROP CONSTRAINT cname;`
Modify referential action	`ALTER TABLE table_name ADD FOREIGN KEY (A) REFERENCES R(B) ON DELETE CASCADE;`

- ALTER TABLE affects only the table structure, not existing data values.
- Constraint names are required when dropping constraints.
- Some DBMSs restrict dropping or modifying attributes referenced by foreign keys.

CREATE TABLE - CONSTRAINTS

- Giving *names* to constraints: CONSTRAINT
  - This is optional.
  - The name is unique within a particular database schema.
  - It is used to identify a particular constraint in case it must be dropped later and replaced with another one.
  - It is also possible to temporarily defer a constraint until the end of a transaction.

32

Note: Có thể tạm hoãn việc kiểm tra ràng buộc cho đến cuối 1 transaction.

DROP Statement

- DROP TABLE:

  DROP TABLE *Dependent* CASCADE;

  - CASCADE: all such constraints, views, and other elements that reference the table are dropped automatically from the schema along with the table itself.

  DROP TABLE *Dependent* RESTRICT;

  - RESTRICT: dropped if it is not referenced in any constraints, views, and other elements.

DROP TABLE DEPARTMENT RESTRICT;

Error Code: 1217. Cannot delete or update a parent row: a foreign key constraint fails (MySQL)

35

Note: Alter Table lúc drop attribute/constraint cũng có quy tắc tương tự như vậy.

3. DML Language

SELECT STATEMENT

```
SELECT [DISTINCT | ALL]             { * | columnExpression [AS newName] |
built_inFunction      [ , ...  ]  }
[FROM TableName | ViewName [alias] [ , ...  ]]
[WHERE rowCondition]
[GROUP BY columnList]
[HAVING groupCondition]
[ORDER BY columnList | columnPositionList [ASC | DESC]]
```

Execution Order	**Clause**	**Meaning**
1	FROM	Specifies and joins table(s) or view(s) to be used
2	WHERE	Filters rows based on row-level conditions
3	GROUP BY	Forms groups of rows having the same grouping column values
4	HAVING	Filters groups based on group-level conditions
5	SELECT	Specifies the output attributes to be returned
6	ORDER BY	Specifies the order of the output