CS2102
AY21/22 SEM 1
github/jovyntls

01. DBMS

- FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY → LIMIT/OFFSET

Transactions

- **transaction, T** → a finite sequence of database operations
- 4 properties of a transaction: **ACID** properties

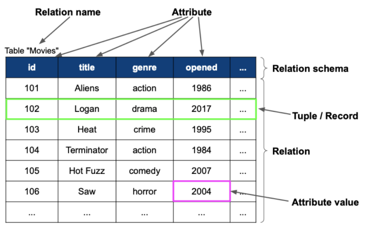
ACID properties

1. **Atomicity** → either all effects of T are reflected in the database, or none
2. **Consistency** → the execution of T guarantees to yield a *correct state* of the DB
3. **Isolation** → execution of T is *isolated* from the effects of concurrent transactions
4. **Durability** → after the commit of T, its effects are *permanent* in case of failures

Serializability

- Requirement for Concurrent Execution: **serializable transaction execution**
  - concurrency: to optimise performance
- (concurrent execution of a set of transactions is) **serializable** → execution is equivalent to some serial execution of the same set of transactions
  - ensures **integrity** of data
- **equivalent** → they have the same *effect* on the data

01-1. RELATIONAL MODEL



id	title	genre	opened	...
101	Aliens	action	1986	...
102	Logan	drama	2017	...
103	Heat	crime	1995	...
104	Terminator	action	1984	...
105	Hot Fuzz	comedy	2007	...
106	Saw	horror	2004	...
...	...	...	...	...

- **relation schema** → defines a relation
  - specifies **attributes** and data constraints
- **relational database schema** → set of relation schemas + data constraints
  - TableName(col_1, col_2, col_3) with dom(col_1) = {x, y, z}
- **domain** → a set of *atomic* values e.g. dom(course) = {cs2102, cs2030, cs2040}
  - $A_i, dom(A_i) =$ set of possible values for A_i
  - for all value v of attribute $A_i, v \in \{dom(A_i) \cup \{null\}\}$
- **relation** → a set of *tuples*
  - $R(A_1, A_2, \dots, A_n)$: relation schema with name R and n attributes $A_1, A_2, \dots, A_n$
  - each instance of schema R is a relation which is a subset of $\{(a_1, a_2, \dots, a_n) \mid a_i \in dom(A_i) \cup \{null\}\}$

Data Integrity

- **integrity constraint** → condition that restricts what constitutes valid data
- **structural** → (integrity) inherent to the data model

Key Constraints

- **superkey** → subset of attributes that *uniquely* identifies a tuple in a relation
- **key** → superkey that is also **minimal** - no proper subset of the key is a superkey
- **candidate keys** → set of all keys for a relation
- **primary key** → selected candidate key for a relation
  - cannot be `null`

Foreign Key Constraints

- **foreign key** → subset of attributes of relation A if it refers to the *primary key* in a relation B.
- Each foreign key in a relation must:
  1. appear as a **primary key** in the referenced relation, OR:
  2. be a `null` value

02. RELATIONAL ALGEBRA

02-1.1 UNARY OPERATORS

Selection, σ_c

- $\sigma_c(R)$ → select all tuples from R satisfying condition c.
  - $\forall$ tuple $t \in R,\ t \in \sigma_c(R) \iff c$ evaluates true on t
  - input and output relation have the same schema
- **selection condition** →
  - a *boolean expression* of one of the following forms:
    - constant selection - attribute **op** constant
    - attribute selection - attribute$_1$ **op** attribute$_2$
    - expr$_1 \wedge$ expr$_2$; expr$_1 \vee$ expr$_2$; $\neg$ expr ; (expr)
  - with **op** $\in \{=, <>, <, \leq, \geq, >\}$
    - **operator precedence**: (), **op**, $\neg, \wedge, \vee$

Projection, π_ℓ

- $\pi_\ell(R)$ → projects all attributes of a given **relation** specified in list ℓ
  - duplicates removed from output relation!
  - **order** of attributes matters!

Renaming, ρ_ℓ

- $\rho_\ell(R)$ → renames the attributes of a relation R (with schema $R(A_1, A_2, \dots, A_n)$)
- 2 possible formats for ℓ
  - ℓ is the new *schema* in terms of the new attribute names
    - $\ell = (B_1, B_2, \dots, B_n)$
    - $B_i = A_i$ if attribute A_i does not get renamed
  - ℓ is a list of attribute renamings of the form: $B_i \leftarrow A_i, \dots, B_k \leftarrow A_k$
    - each $B_j \leftarrow A_j$ renames attribute A_j to attribute B_j
    - order of renaming doesn't matter

02-1.2 SET OPERATORS

- **union** → $R \cup S$ returns a relation w/ all tuples in both R **or** S
- **intersection** → $R \cap S$ ... all tuples in both R **and** S
- **set difference** → $R - S$ ... all the tuples in R **but not in** S

! for all set operators: R and S must be **union-compatible**

Union Compatibility

- two relations R and S are **union-compatible** → if
  - R and S have the same number of attributes; and
  - corresponding attributes have the *same or compatible domains*
- note: R and S do not have to use the same attribute names

02-1.3 CROSS PRODUCT

- **cross product** → given two relations $R(A, B, C)$ and $S(X, Y)$, $R \times S$ returns a relation with schema (A, B, C, X, Y) defined as $R \times S = \{(a, b, c, x, y) \mid (a, b, c) \in R, (x, y) \in S\}$
- **size** of cross product $= |R| * |S|$

02-2. JOIN OPERATORS

Inner Joins

- eliminate all tuples that do not satisfy a matching criteria (i.e. **attribute selection**)
- **θ-join** → (of two relations R and S) $R \bowtie_\theta S = \sigma_\theta(R \times S)$
- **Equi Join** $\bowtie$ → special case of θ-join defined over the **equality** operator ($=$) only
- **Natural Join** $\bowtie$ → performed over all attributes R and S have in common
  - the natural join (of two relations R and S) is defined as $R \bowtie S = \pi_\ell(R \bowtie_c \rho_{b_i \leftarrow a_i, \dots, b_k \leftarrow a_k}(S))$
    - $A = \{a_i, \dots, a_k\} =$ the set of attributes that R and S have in common
    - $c = ((a_i = b_i) \wedge \dots \wedge (a_k = b_k))$
    - $\ell =$ list of all attributes of R + list of all attributes in S that are **not in** A
  - output relation contains the common attributes of R and S only *once*

Outer Joins

- **dangling tuples** → tuples in R or S that do not match with tuples in the other relation
  - ***dangle***$(R \bowtie_\theta S)$ → set of dangling tuples in R wrt to $R \bowtie_\theta S$ (missing attribute values are padded with `null`)
    - *dangle*$(R \bowtie_\theta S) \subseteq R$
  - always removed by inner joins, kept by outer joins
- $null(R)$ → n-component **tuple** of `null` values where n is the number of attributes of R

Definitions

- **left outer join** → $R ⟕_\theta S$ $= R \bowtie_\theta S \cup ($*dangle*$(R \bowtie_\theta S) \times \{$*null*$(S)\})$
- **right outer join** → $R ⟖_\theta S$ $= R \bowtie_\theta S \cup (\{$*null*$(R)\} \times$ *dangle*$(S \bowtie_\theta R))$
- **full outer join** → $R ⟗_\theta S$ $= R \bowtie_\theta S \cup ($*dangle*$(R \bowtie_\theta S) \times \{$*null*$(S)\})$ $\cup (\{$*null*$(R)\} \times$ *dangle*$(S \bowtie_\theta R))$

Natural Outer Joins

- natural left/right/full outer join: $R ⟕ S$ / $R ⟖ S$ / $R ⟗ S$
- output relation contains the common attributes of R and S only once

Handling NULLs

- **comparison** operation with `null` ⇒ *unknown*
- **arithmetic** operation with `null` ⇒ `null`

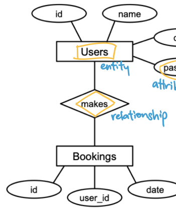
x	y	x<>y	x IS DISTINCT FROM y	x IS NULL
1	1	FALSE	FALSE	FALSE
1	2	TRUE	TRUE	FALSE
null	1	*null*	TRUE	TRUE
null	*null*	*null*	FALSE	TRUE

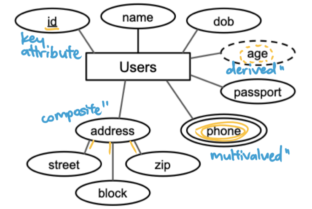
03-1. CONSTRAINTS

- **Not-Null Constraints** violation: $\exists t \in$ Employees where `t.id IS NOT NULL` evaluates to **false**
- **Unique Constraints** violation: For any 2 tuples $t_i, t_k \in R$, $(t_i \cdot A <> t_k \cdot A)$ or $(t_i \cdot B <> t_k \cdot B)$ evaluates to **false**
  - ! `null` rows will NOT violate unique key constraints
- **Primary Key Constraints**: prime attributes cannot be null
  - (entity integrity constraint)
- **Foreign Key Constraints**: each FK in the referencing relation *must*:
  - appear as a PK in the referenced relation, OR
  - be a `null` value
- `R.sid` → `S.id`: `R.sid` is a FK referencing PK `id` in S

04. ENTITY RELATIONSHIP MODEL

- **entity set** → collection of entities of the same type
- **attribute** → specific information describing an entity
  - **key attribute** → uniquely identifies each entity
  - **composite attribute** → composed of multiple other attributes
  - **multivalued attribute** → may comprise more than one value for a given entity
  - **derived attribute** → derived from other attributes

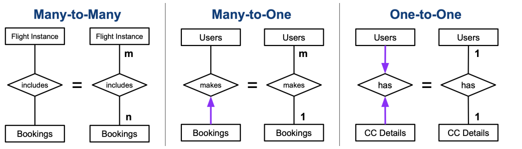




Relationship Sets

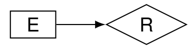
- **degree** → no. of entity roles participating in a relationship
  - an n-ary relationship set involves n entity roles (where n is the degree of the relationship set)

Cardinality Constraints

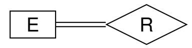


Participation Constraints

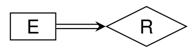
- **partial participation constraint** → participation (of an entity in a relationship) is not mandatory (0 or more)
- **total participation constraint** → participation is mandatory (1 or more)



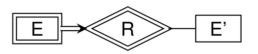
each instance of E participates in **at most one** instance of R



each instance of E participates in **at least one** instance of R



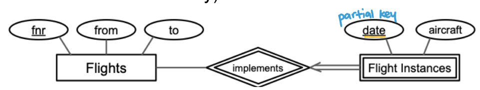
each instance of E participates in **exactly one** instance of R



E is a **weak entity set** with identifying owner E' & identifying relationship set R.

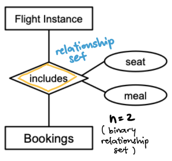
Dependency Constraints

- **weak entity sets** → entity set that does not have its own key
  - can only be uniquely identified through the primary key of its **owner entity**
  - **partial key** → set of attributes that uniquely identifies a weak entity for a given owner entity (identifies the exact instance of a weak entity)

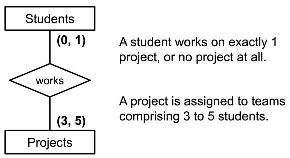


- requirements
  1. many-to-one relationship (identifying relationship) from weak entity set to owner entity set
  2. weak entity set must have **total participation** in identifying relationship

Alternative Representations



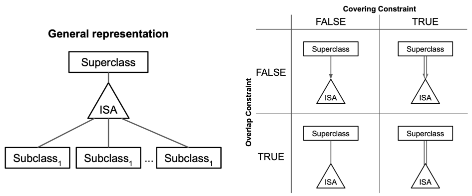
Min/Max notation



04. EXTENDED CONCEPTS

ISA Hierarchy Constraints

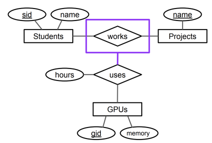
- **overlap contraint** → a superclass entity can belong to **multiple** subclasses
- **covering constraint** → a superclass entity **has to** belong to a subclass



Aggregation

- abstraction that treats relationships as higher-level entities
  - e.g. treating 2 entities + 1 relationship as an entity set

```
CREATE TABLE Uses (
  gid INTEGER,
  sid CHAR(20) ,
  pname VARCHAR(50) ,
  hours NUMERIC,
  PRIMARY KEY (gid , sid , pname) ,
  FOREIGN KEY (gid) REFERENCES GPUs (gid) ,
  FOREIGN KEY (sid , pname) REFERENCES
          works (sid , pname)
);
```



10. FUNCTIONAL DEPENDENCIES

- **normal form** → a definition of minimum requirements in terms of **redundancy**
- an attribute not in any RHS of any FD **must** be in every key
- **prime attribute** → appears in at least one key

Normalisation

Table "Student_Data"

Name	NRIC	Phone	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

⇒

Table "Student_Info"

Name	NRIC	Address
Alice	1234	Jurong East
Bob	5678	Pasir Ris

Table "Student_Contact"

NRIC	Phone
1234	67899876
1234	83838484
5678	98765432

Functional Dependencies

Let $A_1, A_2, \dots, A_m, B_1, B_2, \dots, B_n$ be some attributes

- **uniquely identifies** → $\{A_1 A_2 \dots A_m\} \rightarrow \{B_1 B_2 \dots B_n\}$ whenever 2 tuples have the same values on $A_1 A_2 \dots A_m$, they always have the same values on $B_1 B_2 \dots B_n$
  - "A *uniquely identifies* B": if you know A, then you will know B (but not the other way round)
  - $\{A\} \rightarrow \{B\}$: functional dependency - A determines B

Armstrong's Axioms

1. axiom of **reflexivity**: set → a subset of attributes ($\{A, B\} \rightarrow \{A\}$)
2. axiom of **augmentation**: if $\{A\} \rightarrow \{B\}$, then $\forall C$, $\{AC\} \rightarrow \{BC\}$
3. axiom of **transitivity**: if $\{A\} \rightarrow \{B\}$ and $\{B\} \rightarrow \{C\}$, then $\{A\} \rightarrow \{C\}$

Extended Armstrong's Axioms

- rule of **decomposition**: if $\{A\} \rightarrow \{BC\}$ then $\{A\} \rightarrow \{B\} \wedge \{A\} \rightarrow \{C\}$
- rule of **union**: if $\{A\} \rightarrow \{B\} \wedge \{A\} \rightarrow \{C\}$, then $\{A\} \rightarrow \{BC\}$
- combined: $\{A\} \rightarrow \{BC\} \Leftrightarrow \{A\} \rightarrow \{B\} \wedge \{A\} \rightarrow \{C\}$

Closures

- $B_1 B_2 \dots B_n$ is the closure of $A_1 A_2 \dots A_m$ denoted $\{A_1 A_2 \dots A_m\}^+$

11. BOYCE-CODD NORMAL FORMS (BCNF)

- stronger than 3NF - has fewer redundancies
- may **not** preserve all FDs
  - decomposed table may have no non-trivial & decomposed FDs
  - exists FD that cannot be derived from FDs on R1 and R2
- two **attributes** are **functionally equivalent** → if either one can determine the other

Non-Trivial and Decomposed FD

- **non-trivial** → $\{A\} \rightarrow \{B\}$ where $\{A\} \not\subseteq \{B\}$
- **decomposed** → $\{A\} \rightarrow \{B\}$ where B is a single attribute

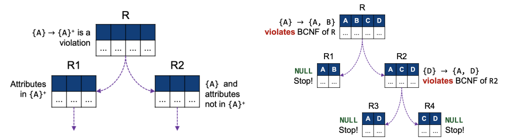
BCNF

- table in **BCNF** → every *non-trivial & decomposed FD* has a *superkey* as its LHS
- **NOT in BCNF** if $\exists$ non-trivial & decomposed FD s.t. its LHS is *NOT* a superkey
- **!** a table with exactly one or two attributes is always in BCNF!
- advantages
  - ✓ no update/deletion/insertion anomalies
  - ✓ small redundancies
  - ✓ **lossless join** - original table can always be reconstructed from decomposed tables (natural join). $\Rightarrow \quad R = R1 \bowtie R2 = \pi_{R1}(R) \bowtie \pi_{R2}(R)$
- **lossless decomposition** → $\{R_1, R_2\}$ is lossless if $R_1 \cap R_2$ is a superkey of R_1 or R_2
  - $R_1 \cap R_2$ uniquely identifies all the attributes in R_1 or R_2
  - closure of $R_1 \cap R_2 = R_1$ or R_2

Normalisation

- normalisation algorithm:
  - for a BCNF-violating FD $\{A\} \rightarrow \{A\}^+$, create tables
    - $R1(\{A\}^+)$ containing the superkey, and
    - $R2\left(\{A\} \cup (R - \{A\}^+)\right)$
- if table does not contain all attributes:
  1. compute closure of each subset of the table's attributes
  2. remove RHS attributes not in the table

! *implicit* functional dependencies should be checked too! (because explicit FDs may not apply to R2 when R2 is missing attributes)



e.g. $R(A, B, C, D)$ with $BC \rightarrow D$, $D \rightarrow A$, $A \rightarrow B$

3NF (THIRD NORMAL FORM)

- will preserve all FDs
- relaxed form of BCNF
  - satisfies BCNF ⇒ satisfies 3NF
  - violates 3NF ⇒ violates BCNF

Functional Dependency Equivalence

let F1 and F2 be sets of FDs.

- **equivalence** → F1 is equivalent to F2 ($F1 \equiv F2$) ⇔
  - $F2 \vdash F1$: every FD in F1 can be derived from F2
  - $F1 \dashv F2$: every FD in F2 can be derived from F1

3NF

- a table is in **3NF** → if every *non-trivial* & *decomposed* FD:
  - its LHS is a **superkey**, OR
  - its RHS is a **prime attribute** (any attribute in any key)
- if all attributes are prime attributes, the table is in 3NF

Minimal Basis

- minimal basis, F_b: **simplified** → 4 conditions
  1. equivalence: $F \equiv F_b$ (every FD in F_b can be derived from F and vice versa)
  2. every FD in F_b is non-trivial and decomposed
  3. $\forall$ FD in F_b, none of the LHS attributes are redundant
  4. no FD in F_b is redundant
- **redundant** → can be removed without affecting the original FD (i.e. $F \equiv F_{b*}$ where F_{b*} is formed by removing the attribute)

to obtain a minimal basis

1. ensure equivalence
2. transform FDs to non-trivial and decomposed
3. remove redundant attributes
   - 3.1. for an FD {A} → B for a **set** of attributes A, for an attribute C in A,
   - 3.2. compute $\{A - C\}^+$ using F
   - 3.3. if $B \in \{A - C\}^+$, then we can remove C
4. remove redundant FDs
   - 4.1. try removing and check for equivalence

3NF Synthesis

for table R and a set of FDs F,

1. derive *minimal basis* F_b of F
2. from the minimal basis, combine the FDs for which the LHS is the same
3. create a table for each FDs remained in the minimal basis after union
4. if **none** of the tables contain a key for the original table R, create a table containing a key of R