Database and Big Data(2019)

Week 4, S2: Normal Forms

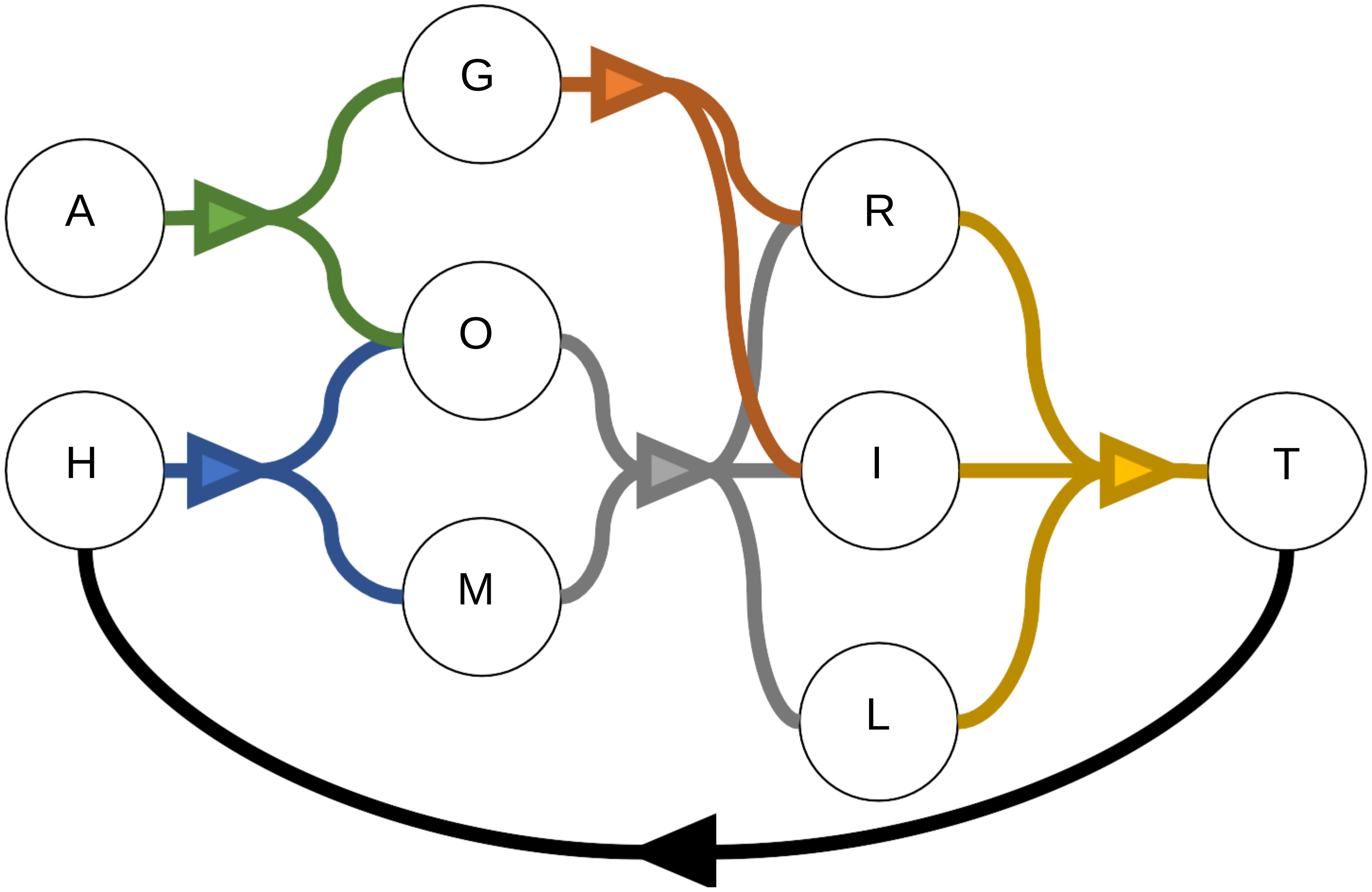
Cyrille Jegourel

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

What will we see today?

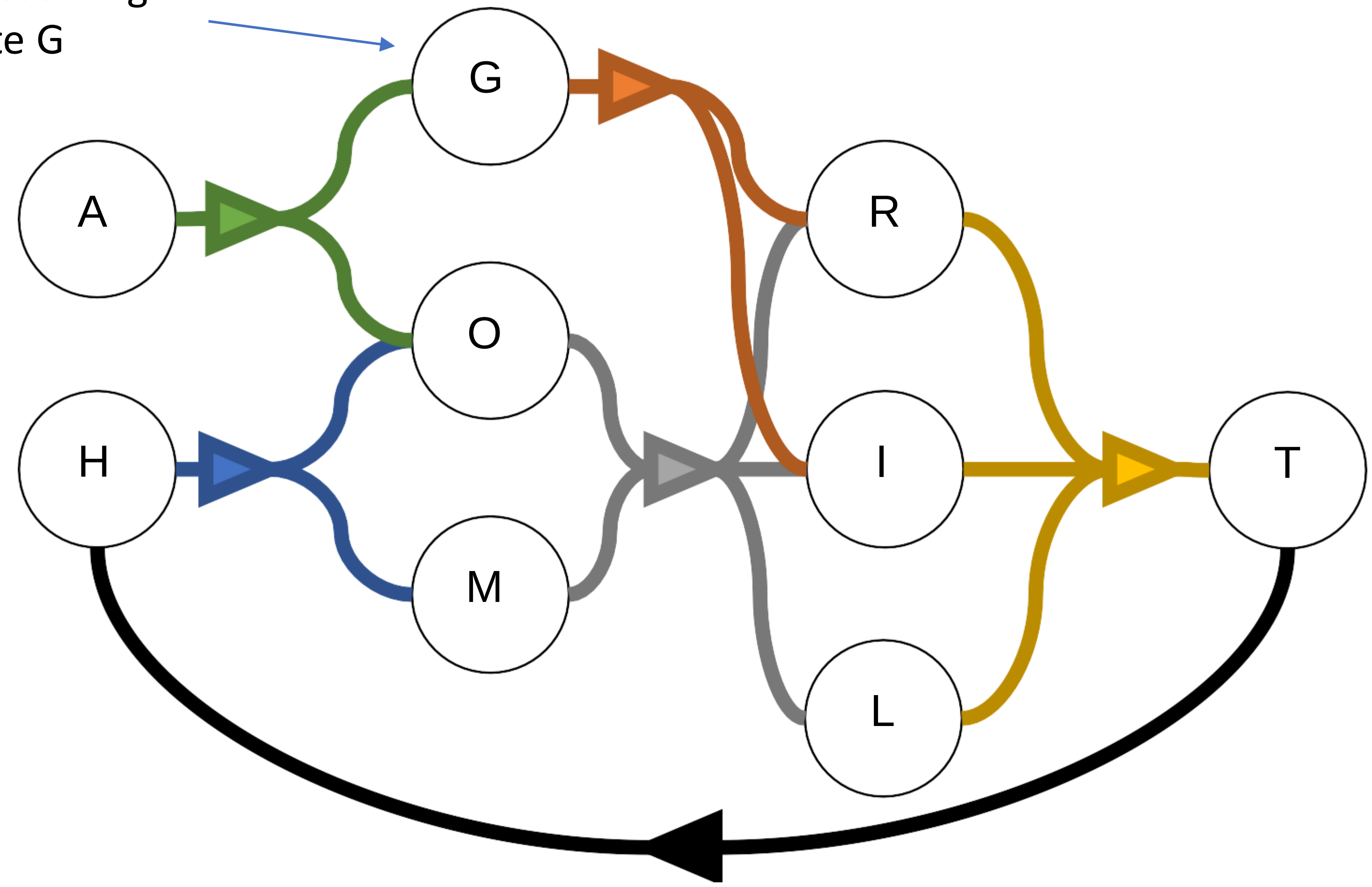
- Visualization of FDs
- Normalization: 1NF, 2NF, 3NF, BCNF, 4NF…
- Losslessness
- Chase method to verify the losslessness of a decomposition.

Visualization of FDs: directed hypergraphs

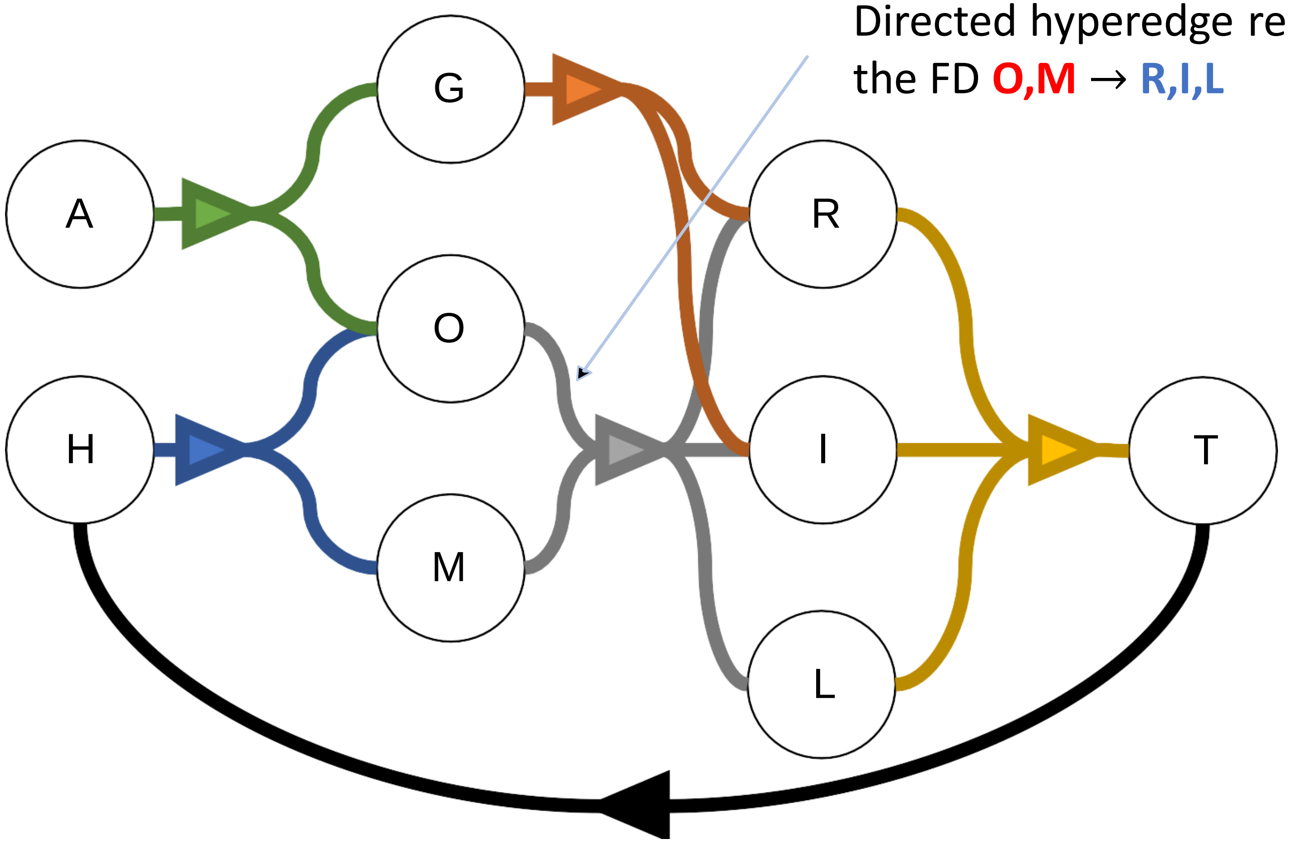


Visualization of FDs: directed hypergraphs

Node representing
an attribute G



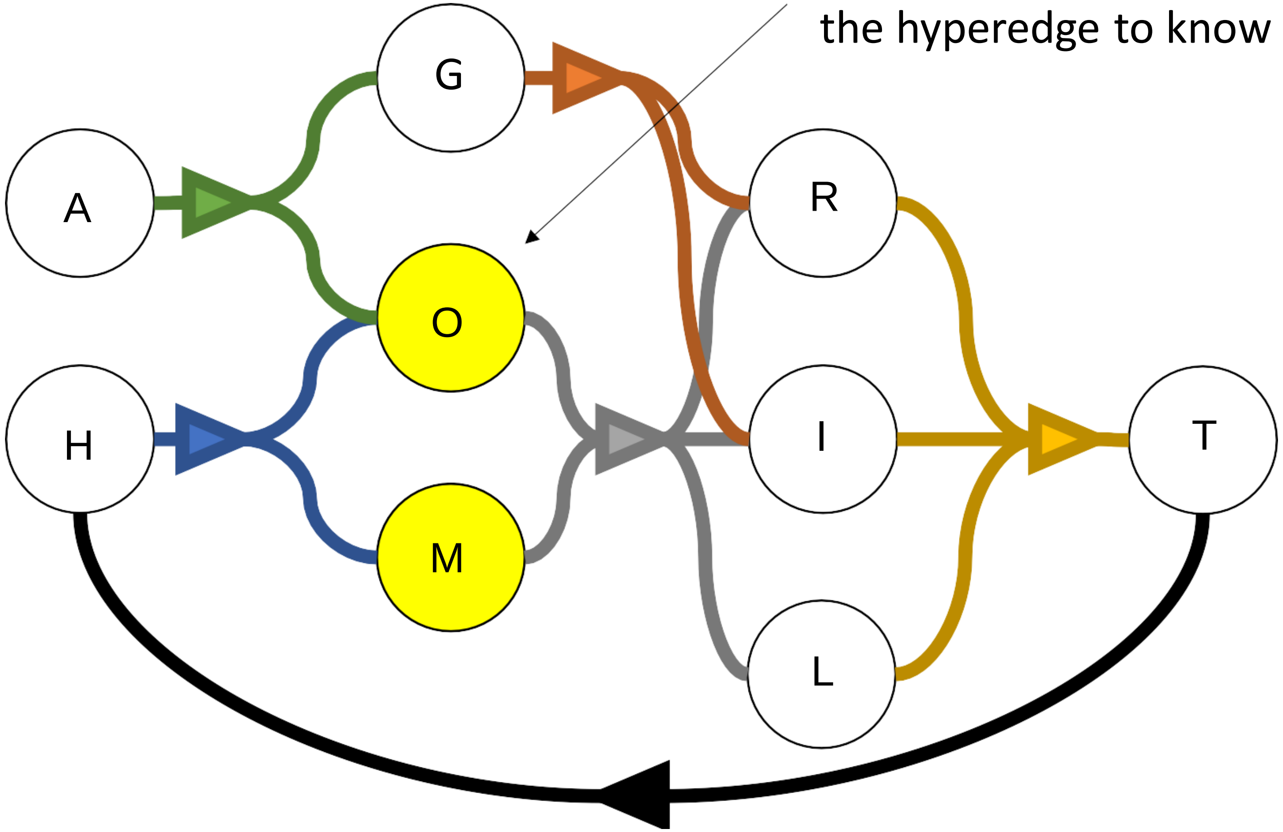
Visualization of FDs: directed hypergraphs



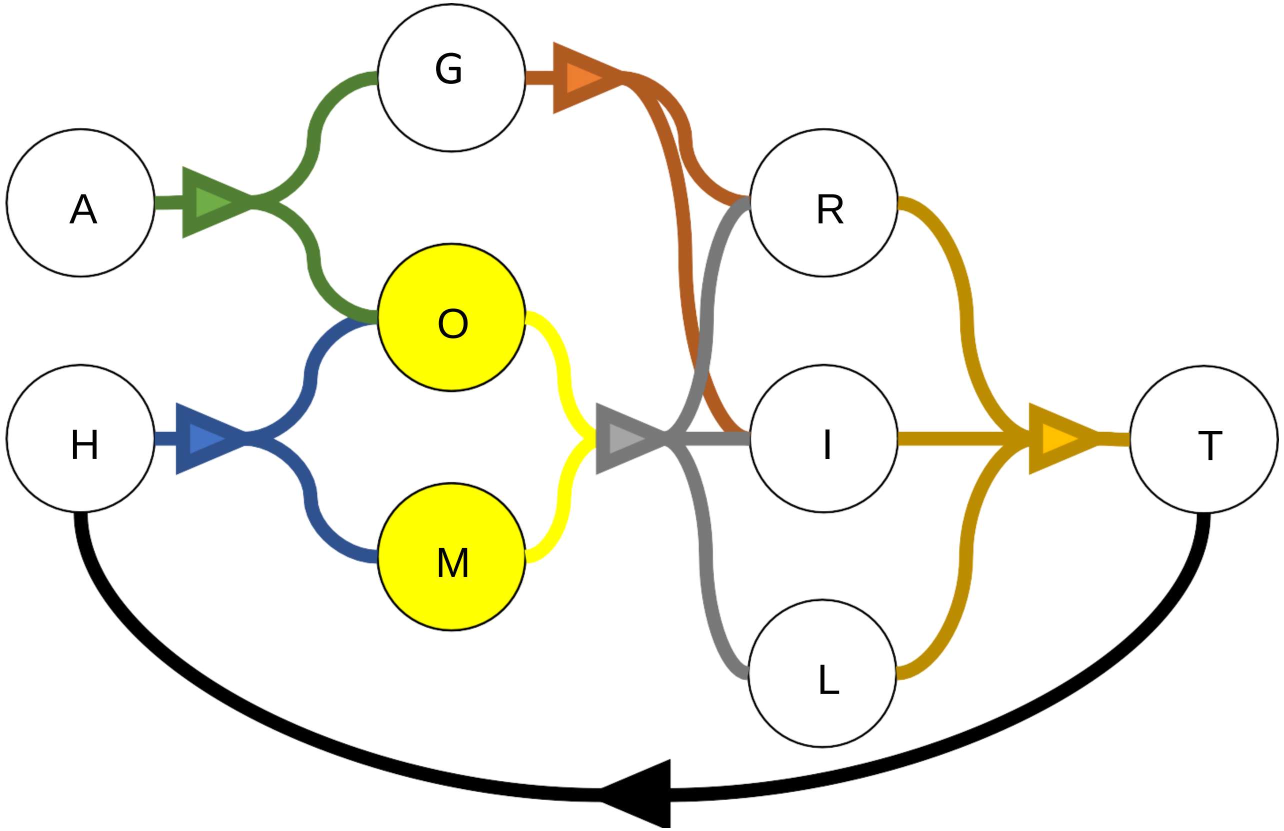
Directed hyperedge representing the FD **O,M** → **R,I,L**

Visualization of FDs: directed hypergraphs

If we know **O** and **M**, we can use the hyperedge to know **R,I,L**

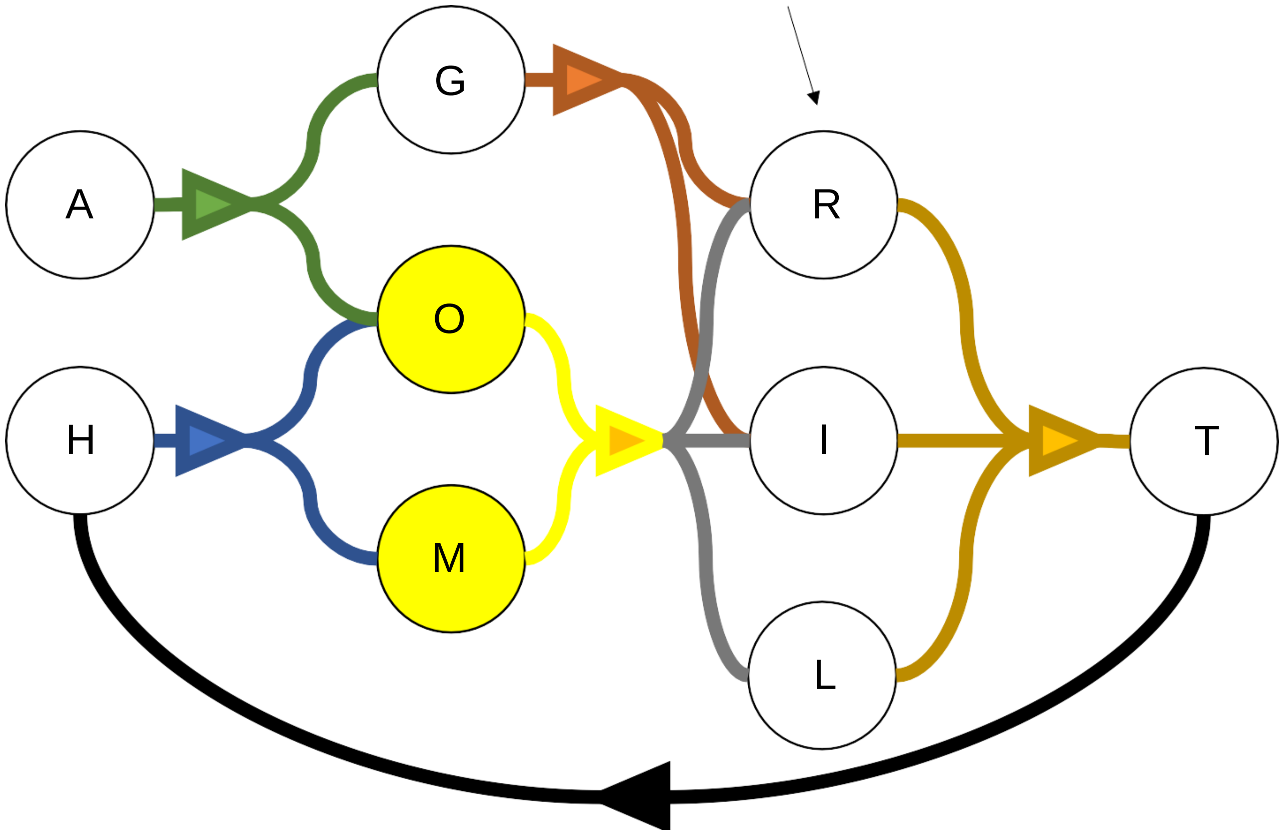


Visualization of FDs: directed hypergraphs



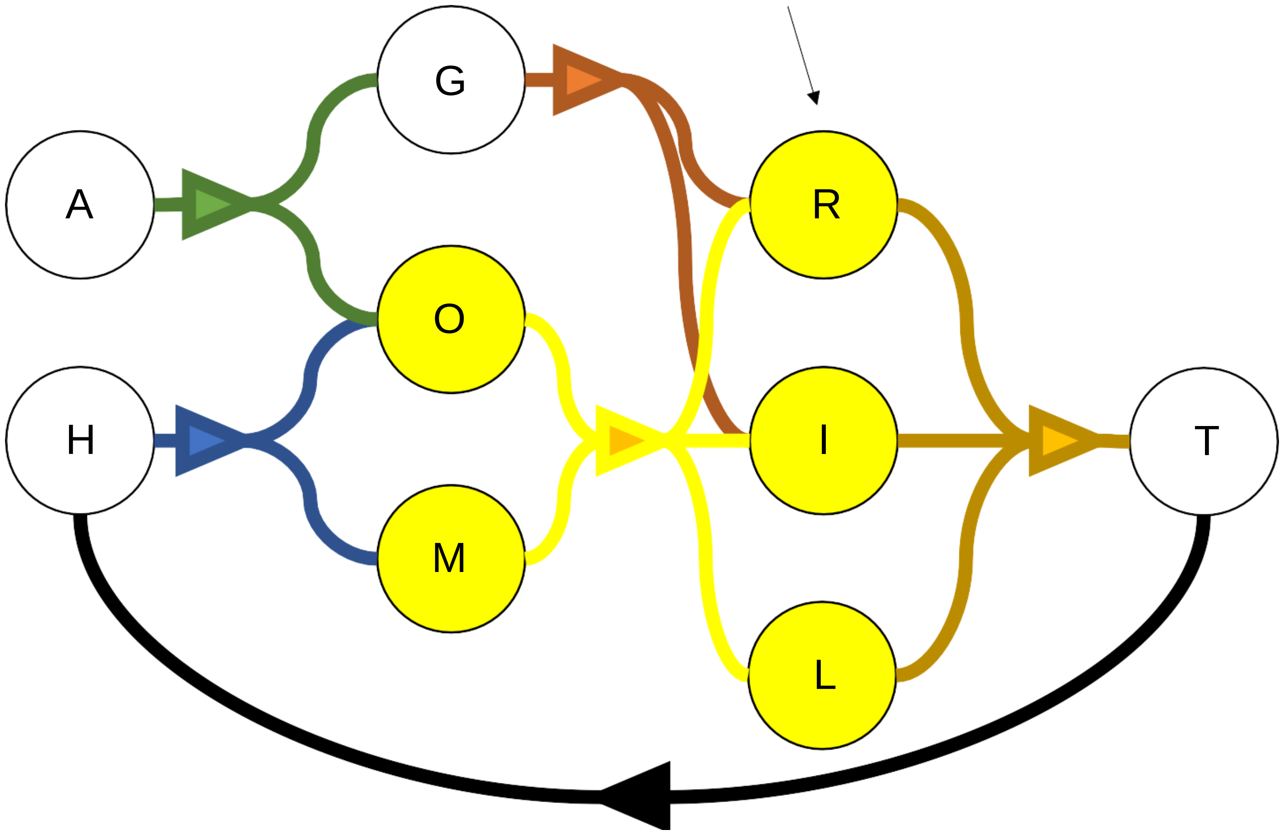
Visualization of FDs: directed hypergraphs

If we know O and M we can use the hyperedge to know R, I, and L



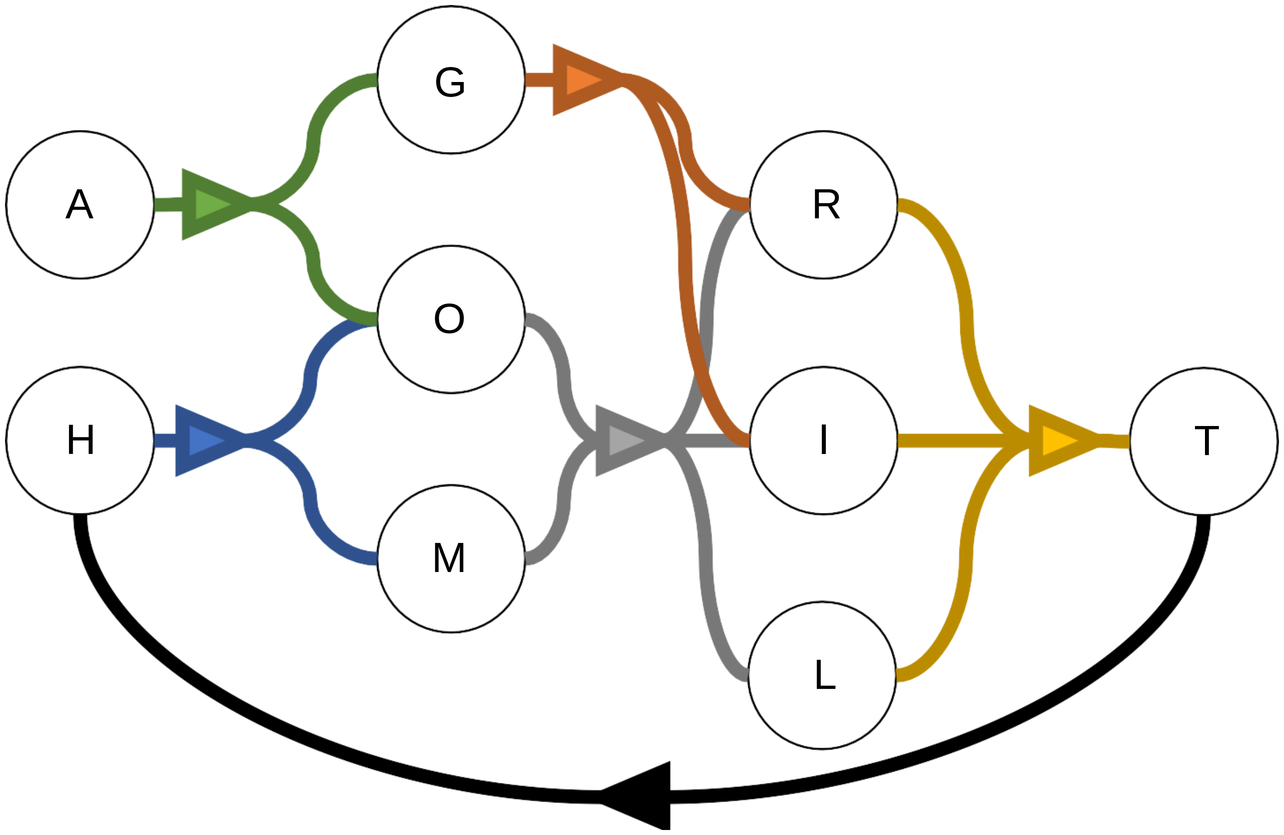
Visualization of FDs: directed hypergraphs

If we know O and M we can use the hyperedge to know R, I, and L



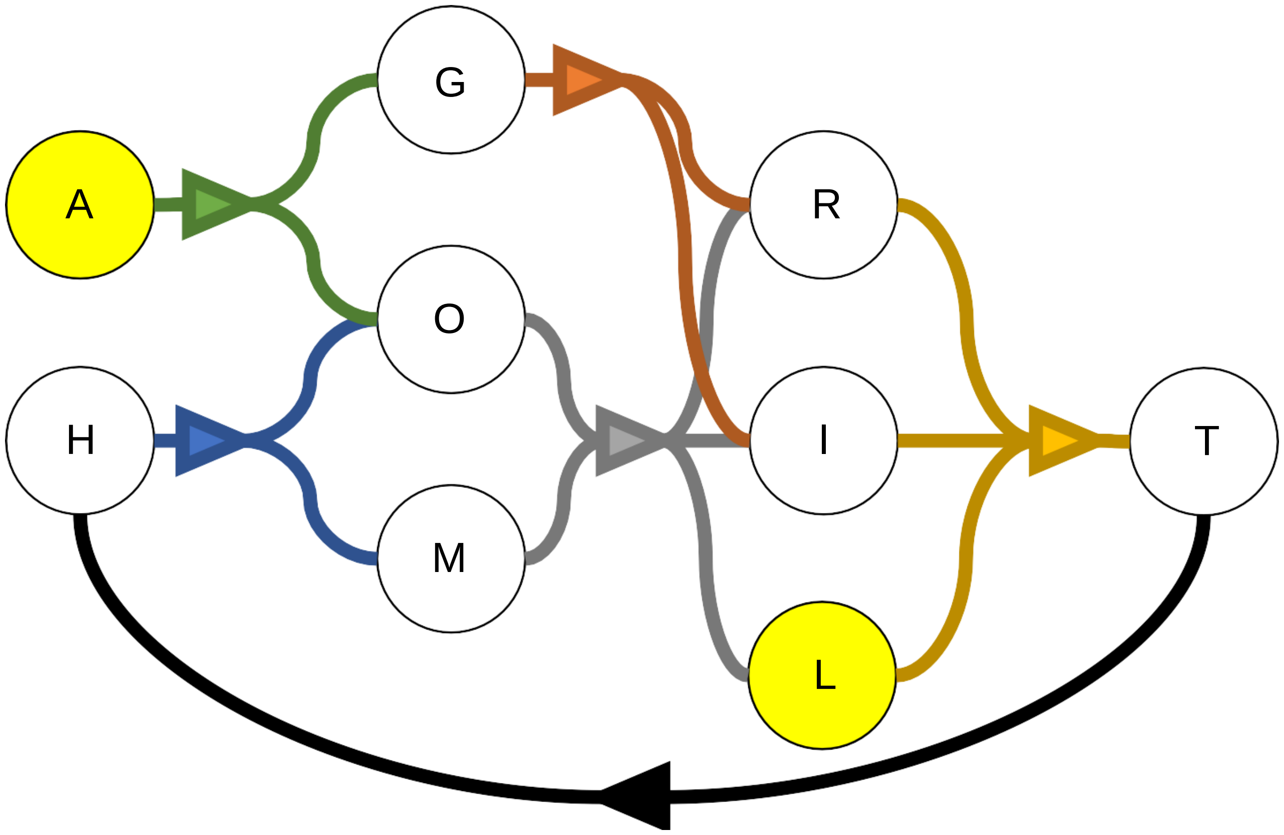
Visualization of FDs: directed hypergraphs

Compute the closure $\{A, L\}^+$



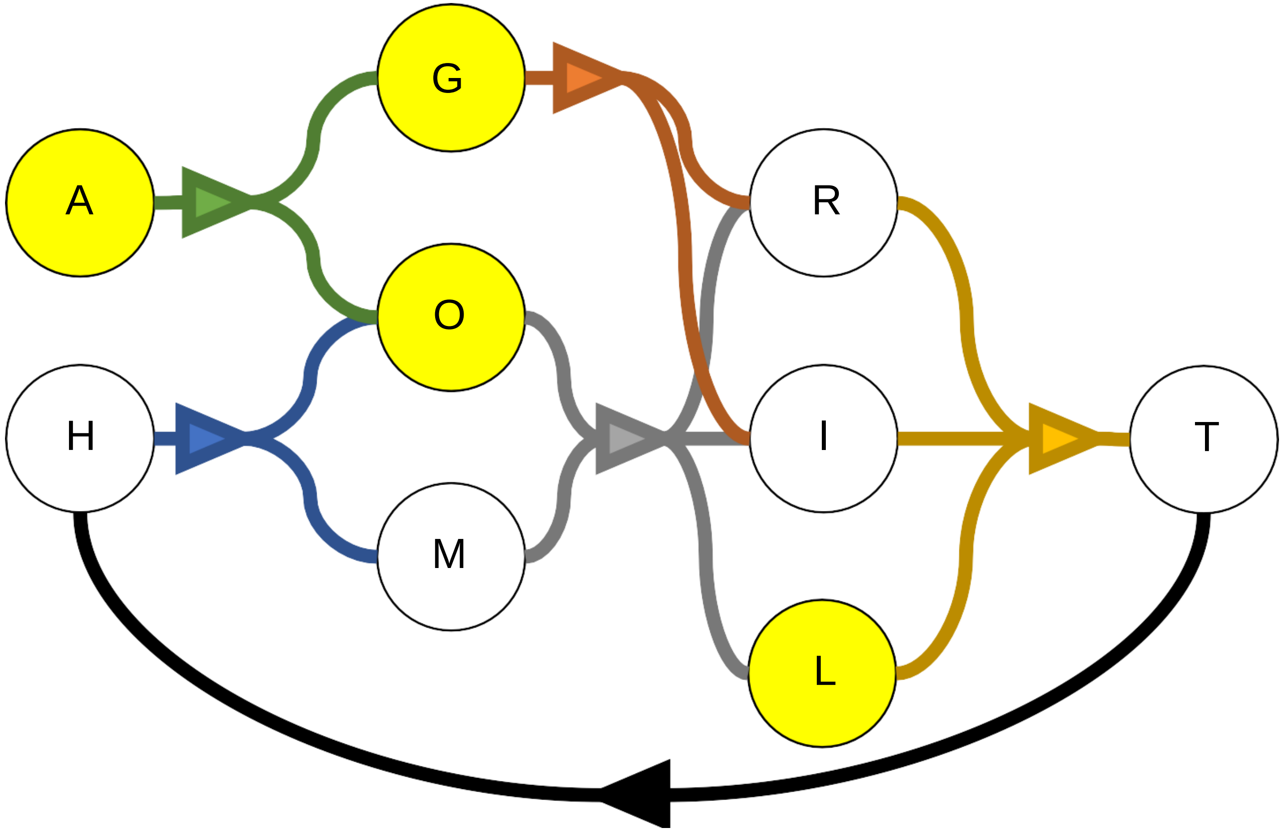
Visualization of FDs: directed hypergraphs

$$\{A, L\}^+ = \{A, L, \ldots\}$$



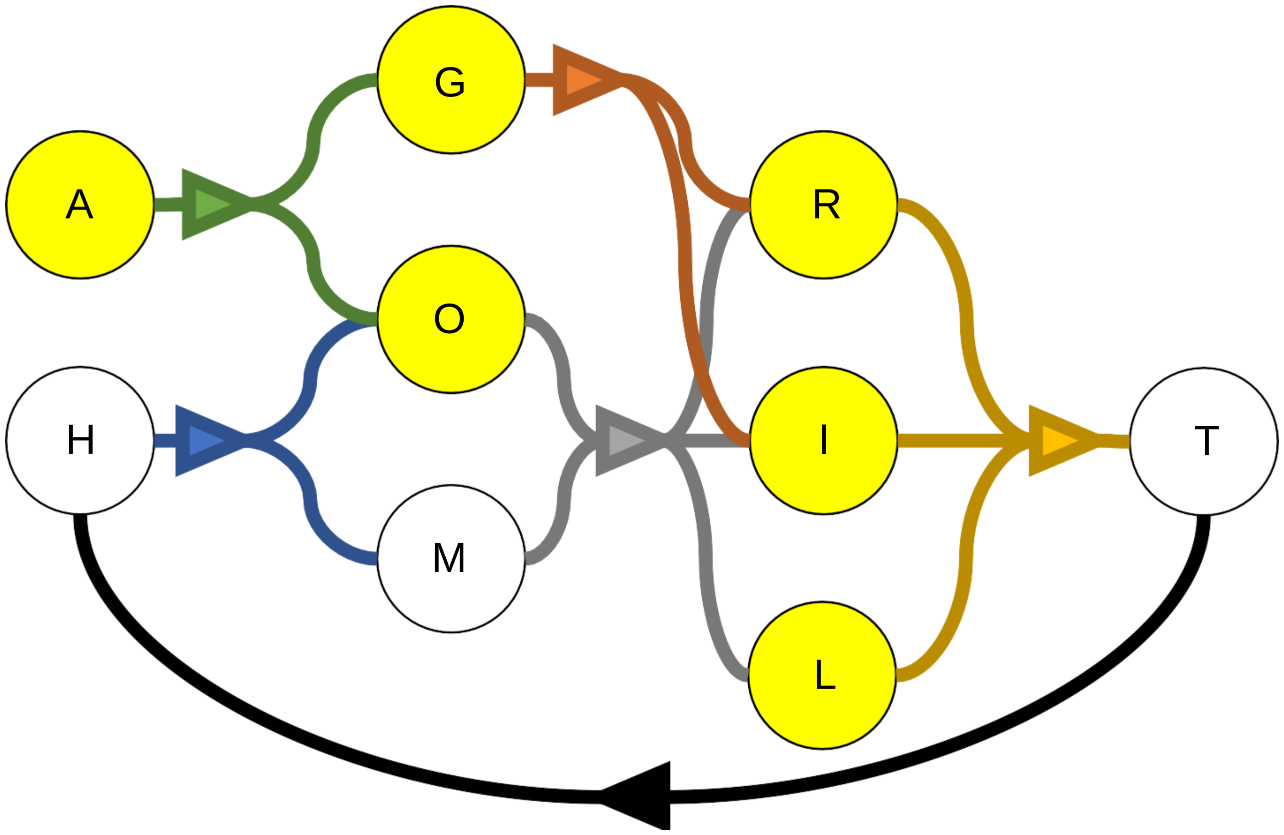
Visualization of FDs: directed hypergraphs

$$\{A, L\}^+ = \{A, L, G, O, \ldots\}$$



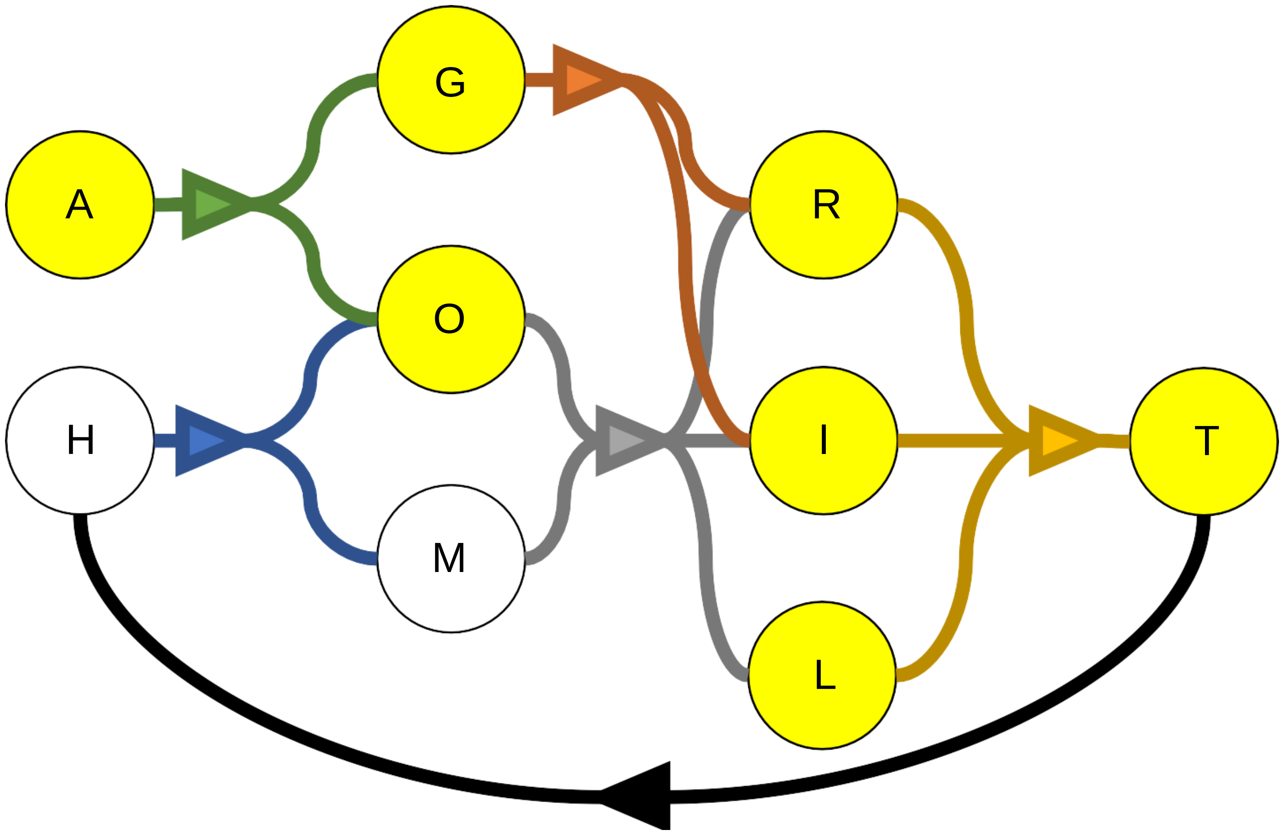
Visualization of FDs: directed hypergraphs

$$\{A, L\}^+ = \{A, L, G, O, R, I \ldots\}$$



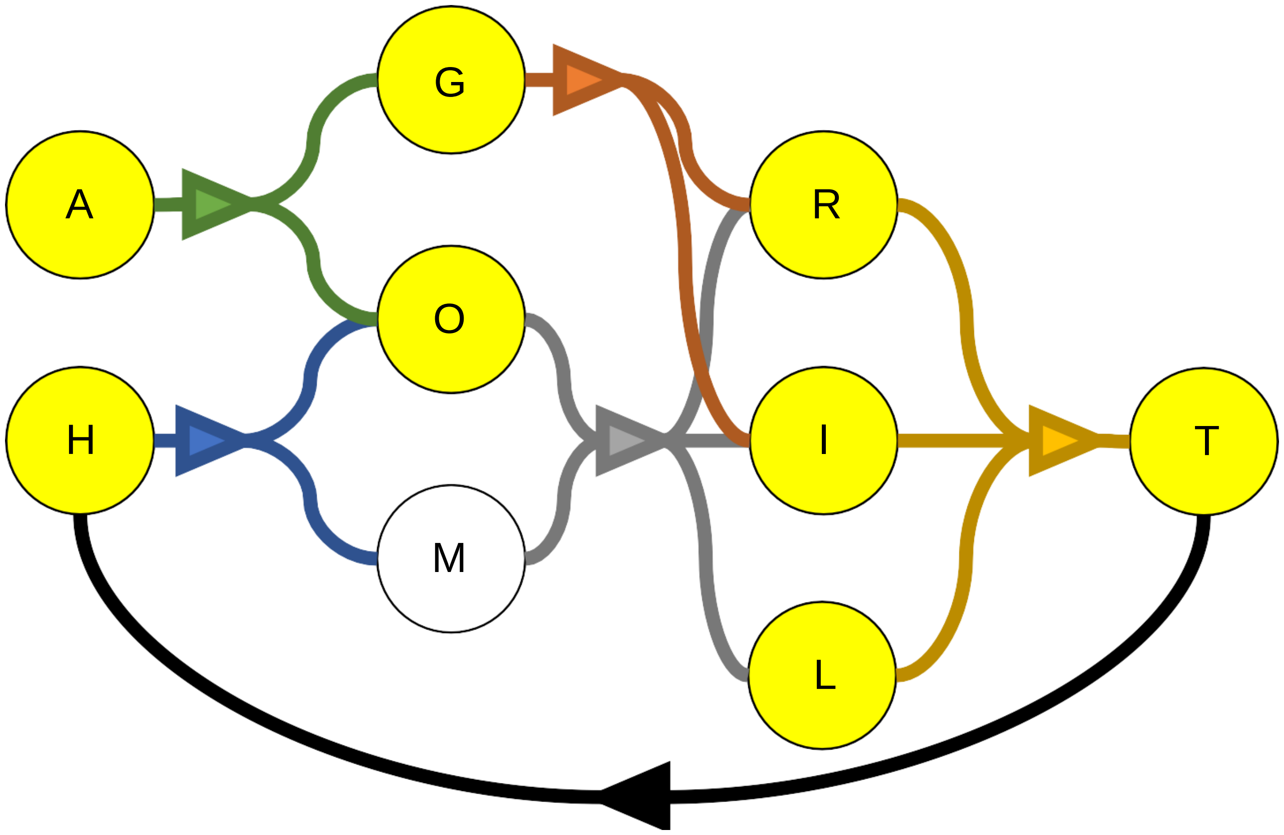
Visualization of FDs: directed hypergraphs

$$\{A, L\}^+ = \{A, L, G, O, R, I, T \ldots\}$$



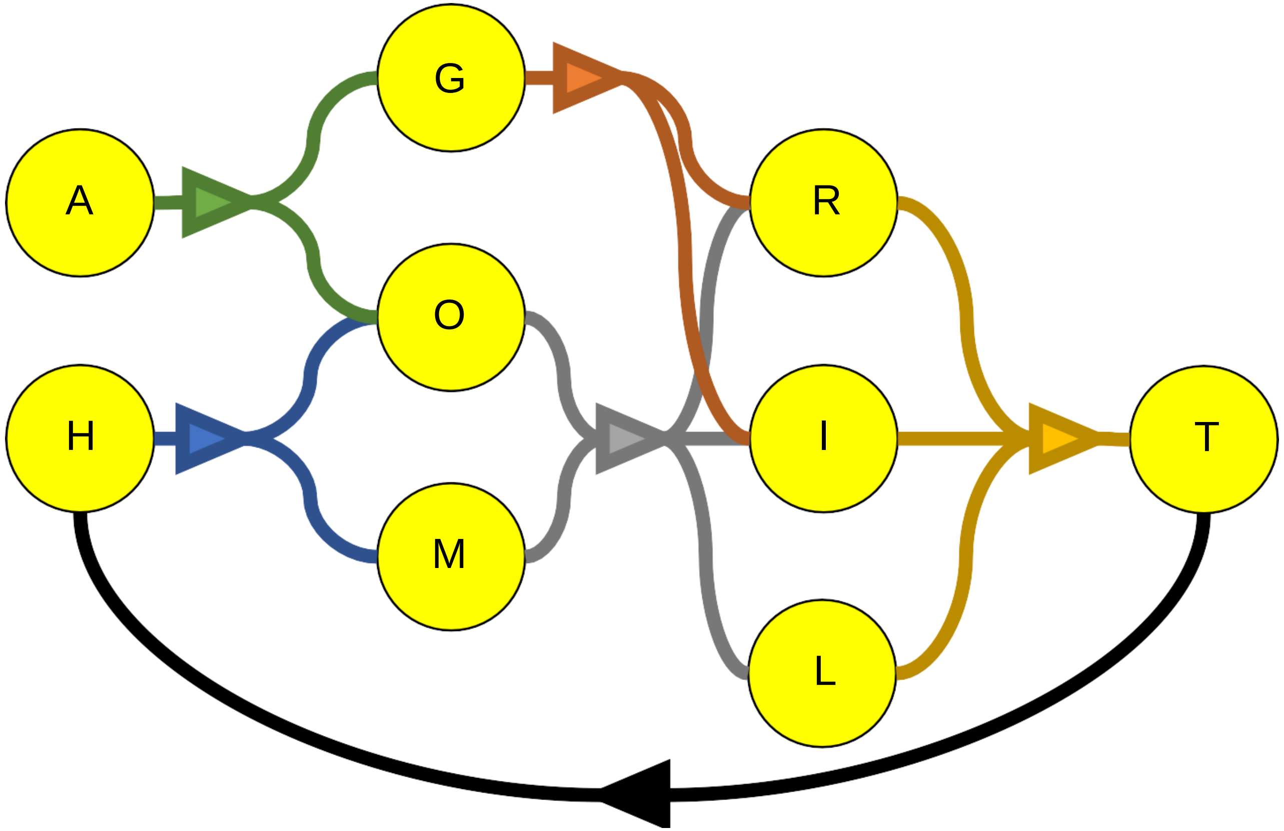
Visualization of FDs: directed hypergraphs

$$\{A, L\}^+ = \{A, L, G, O, R, I, T, H \ \ldots\}$$



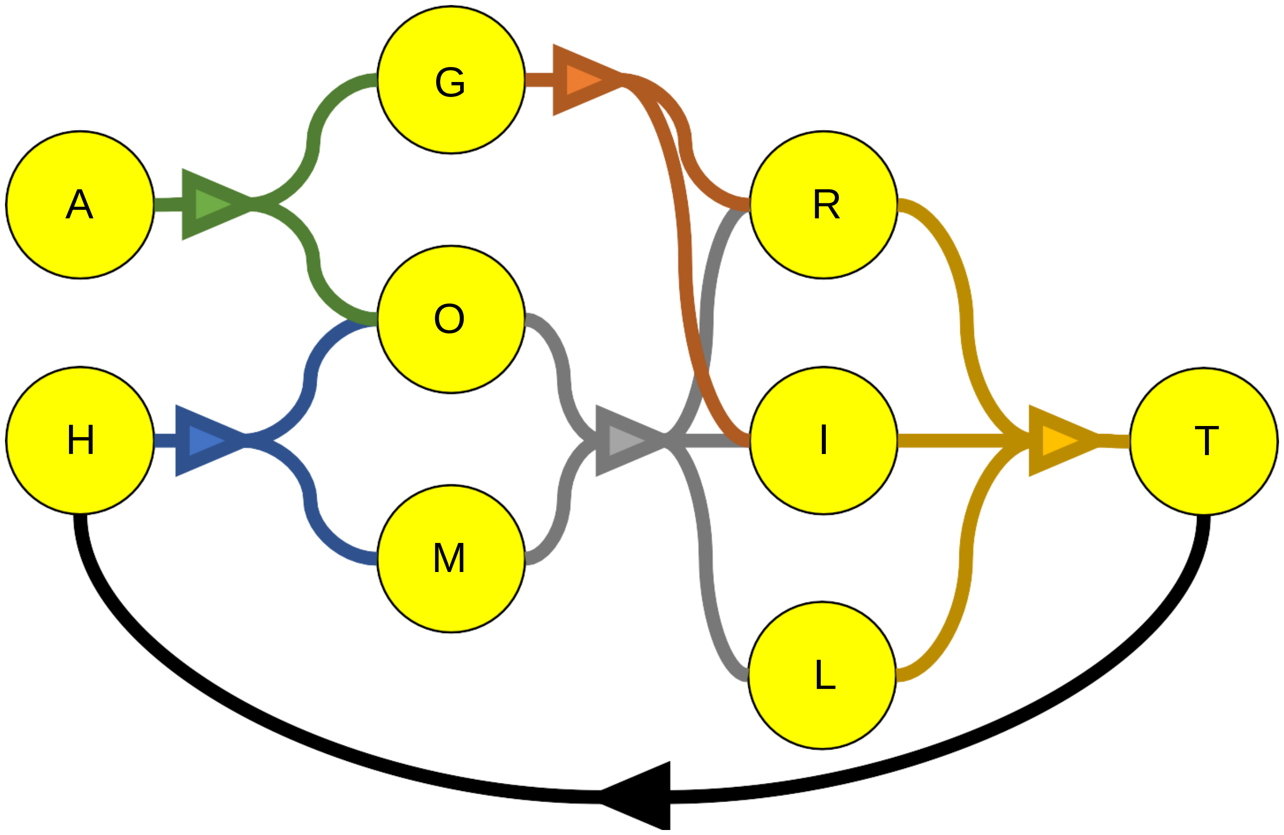
Visualization of FDs: directed hypergraphs

$$\{A, L\}^+ = \{A, L, G, O, R, I, T, H, M\}$$



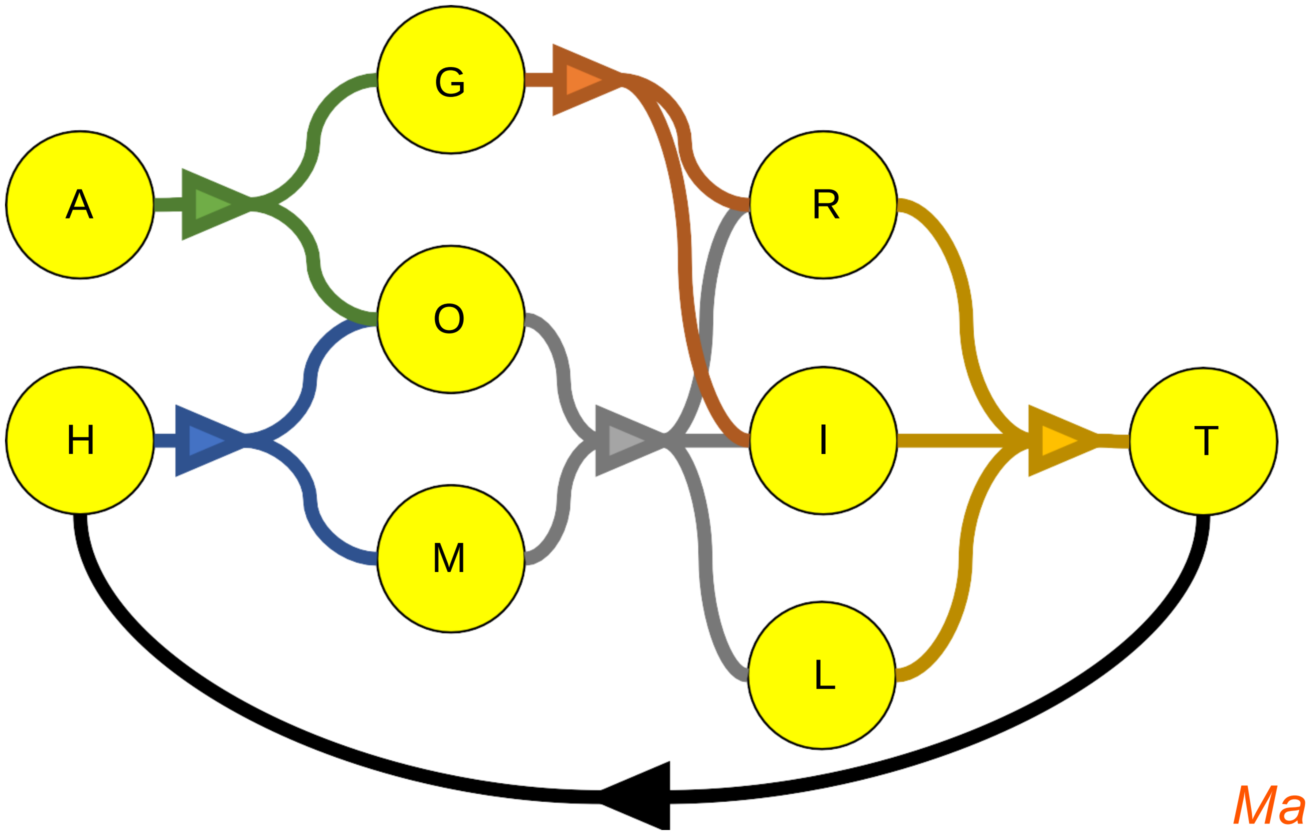
Visualization of FDs: directed hypergraphs

$\{A, L\}$ determines all attributes



Visualization of FDs: directed hypergraphs

$\{A, L\}$ is a (super)key!



Many other super keys

What is normalization?

Normalization is a technique for organizing the data into multiple related tables, to “minimize” data redundancy.

What is data redundancy?

sid	name	pillar	hod	office_tel
123	Agus	ISTD	Mr. Tan	53337
456	Bron	ISTD	Mr. Tan	53337
789	Hannah	ISTD	Mr. Tan	53337
012	Dewi	ISTD	Mr. Tan	53337

Repetition of the same data at multiple places

Why to reduce it?

- Increases the size of the database
- Insert, delete, update problems

Types of normalization

- Normalization can be achieved in several “forms”:
  - 1st normal form (1NF)
  - 2nd normal form (2NF)
  - **3rd normal form (3NF)**
  - **Boyce-Codd normal form (BCNF)**
  - 4th, 5th, 6th normal forms (4NF, 5NF, 6NF)…

1st normal form

- 1st step of the normalization process
- 1NF expects that the table is designed such that it can be easily extended.
- 1NF is mandatory!

How to achieve 1NF?

- 4 basic rules that a table should follow to be in 1NF:
  - Each column should contain atomic values
  - A column should contain values of the same type
  - Each column should have a unique name
  - Order in which data is saved does not matter.

sid, name	pillar
456, Bron	ISTD

1NF: formal definition

1NF

A relation R is in **First Normal Form** if all attribute values are atomic. Attribute values cannot be multivalued.

We call data in 1NF “flat.”

How to achieve 1NF?

sid	name	subject
123	Agus	Database, Computing
456	Bron	Database
789	Hannah	Architecture, Mathematics

Violates 1NF

Split

sid	name	subject
123	Agus	Database
123	Agus	Computing
456	Bron	Database
789	Hannah	Architecture
789	Hannah	Mathematics

1NF: conclusion

- Some values are repeated but values for each column are now atomic for each record/row.
- Using 1NF, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

2nd Normal Form (2NF)

- 2 conditions for a table to be in 2NF:
  - The table has to be in 1NF.
  - It should not have Partial Dependencies.

What is Partial Dependency?

Definition – Full Functional Dependency

In the functional dependency $X \rightarrow A$, an attribute A is **Fully Functionally Dependent** on X if there is no subset Y of X in which $Y \rightarrow A$ holds.

Otherwise, if there is some Y where $Y \rightarrow A$ holds, A is **Partially Dependent** on X.

e.g. **grade** is fully functionally dependent on **sid,course**

2NF: Formal definition

- Definition – Second Normal Form (2NF)
  - A relation R is in **Second Normal Form** if it is in 1NF and all nonprime attributes are fully functionally dependent on the primary key of R.

- Definition – Prime Attribute
  - An attribute is a **Prime Attribute** if it is part of a candidate key, otherwise the attribute is considered **Nonprime**.

Some recall on keys

- Given a super/candidate/primary key, you can fetch any row of data in a table.

student(sid,name,address)

sid	name	address	phone
123	Agus	Bedok	9191
111	Agus	Tampines	2390

{sid,name} is a superkey.
sid and **phone** are candidate keys.
sid is more likely to be the primary key.
If the key is multivalued, we say that the key is composite.

Definition – Prime Attribute

An attribute is a **Prime Attribute** if it is part of a candidate key, otherwise the attribute is considered **Nonprime**

Some recall on keys

- Given a primary key, you can fetch any row of data in a table.

student(sid,name,address,phone)

sid	name	address	phone
123	Agus	Bedok	9191
111	Agus	Tampines	2390

subject(sub_id,sub_name)

sub_id	sub_name
1	Java
2	English
3	Matlab

score(sid,sub_id,marks,teacher)

sid	sub_id	marks	teacher
123	1	70	Oka
123	2	35	Chris
111	1	80	Oka

{sid,sub_id} forms a candidate key.
We choose it as our primary key.

Let’s go back to partial dependencies…

score(sid,sub_id,marks,teacher)

sid	sub_id	marks	teacher
123	1	70	Oka
123	2	35	Chris
111	1	80	Oka

{sid,sub_id} forms a candidate key.
We choose it as our primary key.

- In the score table, teacher only depends on the subject (sub_id), not on the student id.
- Partial dependency: an attribute in a table depends on only a part of the primary key and not on the whole key!
- The score table is not in 2NF.

How to get rid of partial dependencies?

subject(sub_id,sub_name)

sub_id	sub_name
1	Java
2	English
3	Matlab

score(sid,sub_id,marks,teacher)

sid	sub_id	marks	teacher
123	1	70	Oka
123	2	35	Chris
111	1	80	Oka

subject(sub_id,sub_name,teacher)

sub_id	sub_name	teacher
1	Java	Oka
2	English	Chris
3	Matlab	Anh

score(sid,sub_id,marks)

sid	sub_id	marks
123	1	70
123	2	35
111	1	80

2NF: conclusion

- 2NF = 1NF + No Partial Dependency.
- Partial Dependency exists when, for a composite primary key, any attribute in the table depends only on a part of the primary key (i.e. not on the complete primary key).
- To get rid of Partial dependency, divide the table, remove the attribute which is causing partial dependency, and move it to some other table where it fits in well.

3rd normal form (3NF)

- 2 conditions for a table to be in 3NF:
  - The table has to be in 2NF.
  - It should not have Transitive Dependencies.

3NF: formal definition

Definition – Third Normal Form (3NF)

A relation R is in **Third Normal Form** if it is in 2NF and if for all non-trivial FDs, $X \rightarrow A$, X is a superkey or A contains only prime attributes

superkey --> prime attributes

What is transitive dependency?

score(sid,sub_id,marks,exam_type,total)

sid	sub_id	marks	exam_type	total
123	1	70	main	100
123	2	35	oral	40
111	1	80	main	100

- Our primary key is **{sid,sub_id}.**
- Here, exam_type depends on the subject and on student's pillar: so, depends on the primary key.
- However, total depends on exam_type. There is transitive dependency!
- **{sid,sub_id}** → **exam_type** → **total**
- The score table is not in 3NF!

How to remove transitive dependency?

- Let **X** be a primary key and **X** → **Y** → **Z** be a transitive dependency.
- A solution is (1) to remove Y and Z from the main table,
- (2) to create a subtable with Y and Z
- (3) to add an attribute A in both tables to join them.

How to remove transitive dependencies?

score(sid,sub_id,marks,exam_type,total)

sid	sub_id	marks	exam_type	total
123	1	70	main	100
123	2	35	oral	40
111	1	80	main	100

score(sid,sub_id,marks,eid)

sid	sub_id	marks	eid
123	1	70	1
123	2	35	2
111	1	80	1

exam(eid,exam_type,total)

eid	exam_type	total
1	main	100
2	oral	40

3NF: conclusion

- 3NF = 2NF + No Transitive Dependency.
- Transitive Dependency exists when an attribute in the table does not depend directly from the primary key but from intermediate attributes.
- To get rid of Partial dependency: remove the attributes which are causing transitive dependency in the main table, create a subtable, add an attribute to join both tables.
- Removing transitive dependency:
  - Amount of data duplication is reduced.
  - Data integrity achieved.

Boyce-Codd normal form (BCNF)

- Upgraded version of 3NF. Sometimes called 3.5 normal form.
- 2 conditions for a table to be in BCNF:
  - The table should be in 3NF.
  - For any dependency **X** → Y, **X** should be a superkey.
- So, if for a dependency **X** → **Y**, **X** is a non-prime attribute and **Y** is a prime attribute, the table is **NOT** in BCNF.

BCNF: formal definition

BCNF

A relation R is in **Boyce-Codd Normal Form (BCNF)** if for every non-trivial dependency, $X \rightarrow A$, X is a superkey.

Equivalently, a relation R is in BCNF if $\forall X$ either $X^+ = X$ or $X^+ = C$ where C is the set of all attributes in R

Examples

- $R(A, B, C)$ with FDs $A \rightarrow B$ and $B \rightarrow C$ BCNF?
- $R(A, B, C)$ with FDs $A \rightarrow BC$ BCNF?
- $R(A, B, C)$ and $S(A, D, E)$ with FDs $A \rightarrow BCDE$ and $E \rightarrow AD$ BCNF?

trivial dependency is A -> A

BCNF: formal definition

BCNF

A relation R is in **Boyce-Codd Normal Form (BCNF)** if for every non-trivial dependency, $X \rightarrow A$, X is a superkey.

Equivalently, a relation R is in BCNF if $\forall X$ either $X^+ = X$ or $X^+ = C$ where C is the set of all attributes in R

Examples

- $R(A, B, C)$ with FDs $A \rightarrow B$ and $B \rightarrow C$ is not in BCNF
- $R(A, B, C)$ with FDs $A \rightarrow BC$ is in BCNF
- $R(A, B, C)$ and $S(A, D, E)$ with FDs $A \rightarrow BCDE$ and $E \rightarrow AD$ is in BCNF

A -> B, B -> C

- { A }+ = { A, B, C }
- { C }+ = { C }
- { B }+ = { B, C } X
Thus, NOT BCNF

A -> BC

- { A }+ = { A, B, C }
- { B }+ = { B }
- { C }+ = { C }
Thus, BCNF

A -> BCDE, E -> AD
In R(A, B, C),

{ A }+ = { A, B, C }
{ B }+ = { B }
{ C }+ = { C }

In S(A, D, E),

{ A }+ = { A, D, E }
{ D }+ = { D }
{ E }+ = { E, A, D }
Thus, BCNF

Recall on dependencies

- **Attribute** → **attribute** — Functional dependency
- **Part of primary key** → **non-prime attribute** — Partial dependency
  Not allowed in 2NF
- **Non-prime attribute** → **non-prime attribute** — Transitive dependency
  Not allowed in 3NF
- **Non-prime attribute** → **prime attribute**

  Is this possible? Not in BCNF.

How can we have Non-prime → prime ?

subject(sid,subject_name,teacher)

sid	subject_name	teacher
123	Java	Oka
123	English	Chris
456	Java	Norman
789	Maths	Cyrille
012	Java	Oka

- Multiple professors teach Java
- sid,subject_name is our primary key:
  sid,subject_name → **teacher**
- But we also have **teacher** → **subject_name**

1-2-3-BC-NF

subject(sid,subject_name,teacher)

sid	subject_name	teacher
123	Java	Oka
123	English	Chris
456	Java	Norman
789	Maths	Cyrille
012	Java	Oka

- All attributes are single-valued: satisfies 1NF.
- We have **sid,subject_name** → **teacher** and **teacher** → **subject_name**. But we don't have **sid** → **teacher** or **subject_name** → **teacher**. No partial dependencies: satisfies 2NF.
- No transitive dependencies as well: satisfies 3NF.
- But because of **teacher** → **subject_name**, the table does not satisfy BCNF.

How to make the table satisfy BCNF?

subject(sid,subject_name,teacher)

sid	subject_name	teacher
123	Java	Oka
123	English	Chris
456	Java	Norman
789	Maths	Cyrille
012	Java	Oka

student(sid,tid)

sid	tid
123	1
123	2
456	3
789	4
012	1

teacher(tid,subject_name,teacher)

tid	subject_name	teacher
1	Java	Oka
2	English	Chris
3	Java	Norman
4	Maths	Cyrille

Decomposition

- **Extract** attributes using **decomposition** (split the schema into smaller parts)
- Here, decomposition means:

Do not forget the common attributes to join the data!

$$R_1(\boxed{A_1, \dots, A_n}, B_1, \dots, B_m)$$

$$R(A_1, \dots, A_n, B_1, \dots, B_m, C_1, \dots, C_k)$$

tid **sid** **subject_name, teacher**

$$R_2(\boxed{A_1, \dots, A_n}, C_1, \dots, C_k)$$

BCNF decomposition algorithm

$Normalize(R)$
  $C \leftarrow$ the set of all attributes in R
  find X **s.t.** $X^+ \neq X$ **and** $X^+ \neq C$
  if X is not found
  then "R is in BCNF"
  else
    decompose R into $R_1(X^+)$ and $R_2((C - X^+) \cup X)$
    $Normalize(R_1)$
    $Normalize(R_2)$

BCNF decomposition algorithm

$Normalize(R)$
  $C \leftarrow$ the set of all attributes in R
  find X **s.t.** $X^+ \neq X$ **and** $X^+ \neq C$
  if X is not found
  then "R is in BCNF"
  else
    decompose R into $R_1(X^+)$ and $R_2((C - X^+) \cup X)$
    $Normalize(R_1)$
    $Normalize(R_2)$

Determine if R is in BCNF already

Decompose into a relation where X is a superkey

Decompose into a relation with X and attributes X cannot determine

Losslessness

Definition

Lossless Decomposition is a reversible decomposition, i.e. rejoining all decomposed relations will always result exactly with the original data.

This is the opposite of a **Lossy Decomposition**, an irreversible decomposition, where rejoining all decomposed relations may result something other than the original data, specifically with extra tuples.

BCNF: pros and cons

- **Is BCNF decomposition lossless? Yes!**
  - For those who are interested: look at **Heath's theorem.**

- **Does BCNF preserve the FDs? Not necessarily.**
  - Some FDs of the original relation may not be all covered after the decomposition.

BCNF: conclusion

- BCNF = 3NF + No dependency of type non-prime → prime.
- To get rid of non-prime → prime : use the BCNF algorithm.
- BCNF decomposition is lossless.

4th normal form (4NF)

- 2 conditions for a table to satisfy 4NF:
  - It should satisfy BCNF.
  - It should not have multi-valued dependency.

What is multi-valued dependency?

Definition

- If the table has at least 3 columns (**A**,**B**,**C**),
- If, given a functional dependency **A** $\rightarrow$ **B**, for a single value A_1 of attribute **A**, more than one value (e.g. B_1 and B_2) of **B** exist,
- And if **B** and **C** are independent of each other,

the table has a multi-valued dependency.

Multi-valued dependency

student(sid,subject,hobby)

sid	subject	hobby
123	Java	chess
123	English	wine tasting
456	Java	tennis
789	Maths	wushu
012	Java	salsa

student(sid,subject,hobby)

sid	subject	hobby
123	Java	chess
123	Java	wine tasting
123	English	chess
123	English	wine tasting
456	Java	tennis
789	Maths	wushu
012	Java	salsa

Multi-valued dependency leads to unnecessary repetition of data.

How to remove multi-valued dependency?

student(sid,subject,hobby)

sid	subject	hobby
123	Java	chess
123	English	wine tasting
456	Java	tennis
789	Maths	wushu
012	Java	salsa

student(sid,subject)

sid	subject
123	Java
123	English
456	Java
789	Maths
012	Java

student(sid,hobby)

sid	hobby
123	chess
123	wine tasting
456	tennis
789	wushu
012	salsa

4NF and more: conclusion

- 4NF = BCNF + No multi-valued dependency
- Multi-valued dependency can be easily removed by “separating” the independent multi-valued attributes into subtables.
- There exist 5NF and 6NF but they are not necessary (and may even affect performance).
- BCNF is the main standard of database. You should always decompose your database in BCNF!

Is a decomposition lossless?

- We need a way to verify if a decomposition is lossless.
- That means to check that joining decompositions $S_1,\ldots,S_n$ equals the original relation R:

$$R = S_1 \bowtie \cdots \bowtie S_n \ ?$$

- Showing $R \subseteq S_1 \bowtie \cdots \bowtie S_n$ is usually simple. Just check if you can naturally join the subtables (need of a joint attribute) and if all the attributes of R are represented.
- Showing $R \supseteq S_1 \bowtie \cdots \bowtie S_n$ is trickier.

Chase method (step-by-step)

You can determine with a chase algorithm if a decomposition is lossless.

1. Generate a tableau of generic tuples (a,b,c,d) representing each schema.
2. Each generic tuple (a,b,c,d) has known values corresponding to the respective projection.
3. Until a row reflects the original generic tuple, continue to chase on FDs (extract more agreements of values)

Chase method: an example

- Let R(A,B,C,D) be a relation with FDs: **A** → **B**, **B** → **C**, **CD** → **A** and S_1(A,D), S_2(A,C) and S_3(B,C,D) be a decomposition of R into 3 (projected) relations.
- We want to prove $R \supseteq S_1 \bowtie S_2 \bowtie S_3$.
- Let's prove that $(a, b, c, d) \in S_1 \bowtie S_2 \bowtie S_3$ implies $(a, b, c, d) \in R$.
- We already know that $(a, d) \in S_1$, $(a, c) \in S_2$ and $(b, c, d) \in S_3$.

1st line corresponds to S1
2nd line to S2, 3rd line to S3

A	B	C	D
a	b1	c1	**d**
a	b2	**c**	d2
a3	**b**	**c**	**d**

Chase method: an example

- Let R(A,B,C,D) be a relation with FDs: **A** → **B**, **B** → **C**, **CD** → **A** and S_1(A,D), S_2(A,C) and S_3(B,C,D) be a decomposition of R into 3 (projected) relations.
- The tableau can be chased by applying the FDs to equate symbols in the tableau. Final tableau with a row that is the same as *(a,b,c,d)* implies that any tuple *(a,b,c,d)* in the join of the projections is a tuple of *R*.

A	B	C	D
a	b1	c1	**d**
a	b2	**c**	d2
a3	**b**	**c**	**d**

Apply **A** → **B**

A	B	C	D
a	b1	c1	**d**
a	b1	**c**	d2
a3	**b**	**c**	**d**

Apply **B** → **C**

A	B	C	D
a	b1	**c**	**d**
a	b1	**c**	d2
a3	**b**	**c**	**d**

When equating two symbols, if both have their own subscript, uniformize

When equating two symbols, if one of them is unsubscripted, make the other be the same.

Chase method: an example

A	B	C	D
a	b1	c	d
a	b1	c	d2
a3	b	c	d

Apply **CD** → A

A	B	C	D
a	b1	c	d
a	b1	c	d2
a	b	c	d

A	B	C	D
a	b	c	d
a	b	c	d
a	b	c	d

When equating two symbols, if one of them is unsubscripted, make the other be the same.

$(a, b, c, d) \in R$

Conclusion

- What have you seen today?
  - Recall about keys, functional dependencies…
  - Normal forms
  - Losslessness
  - Chase method