Information Integration

3NF SYNTHETIS

MEDIATORS

WAREHOUSING

ANSWERING QUERIES USING VIEWS

Functional Dependencies (Recap)

FD X ->Y is an assertion about a relation *R* that whenever two tuples of *R* agree on all the attributes of *X*, then they must also agree on all attributes in set *Y*.

- Say "*X ->Y* holds in *R*."
- Convention: …, *X*, *Y*, *Z* represent sets of attributes; *A*, *B*, *C*,… represent single attributes.
- Convention: no set formers in sets of attributes, just *ABC*, rather than {*A*,*B*,*C*}.

Inferring FD's

We are given FD's X_1 -> A_1, X_2 -> A_2,…, X_n -> A_n , and we want to know whether an FD Y -> B must hold in any relation that satisfies the given FD's.

- Example: If A -> B and B -> C hold, surely A -> C holds, even if we don't say so.

Important for design of good relation schemas.

Closure Test for Inferring FDs

One way to test is to compute the *closure* of Y, denoted Y^+.

Closure Y^+ is the set of attributes that Y functionally determines.

Basis: $Y^+ = Y$.

Induction: Look for an FD's left side X that is a subset of the current Y^+. If the FD is $X \to A$, add A to Y^+.

Example – Closure Test

- Assume *ABCD -> F, ABC -> D, F -> GH, I -> JGH*.
- *What is the closure of ABC, ABC^{+}?*

Example – Closure Test

- Assume *ABCD -> F, ABC -> D, F -> GH, I -> JGH*.
- *What is the closure of ABC, ABC^{+}?*

1. Basis $ABC^{+} = ABC$

Example – Closure Test

- Assume *ABCD -> F, ABC -> D, F -> GH, I -> JGH*.
- *What is the closure of ABC, ABC^{+}?*

1. Basis $ABC^{+} = ABC$
2. $ABC^{+} = ABC^{+}$ union *D*, since ABC is a subset of ABC^{+} and *ABC -> D*

Example – Closure Test

- Assume *ABCD -> F, ABC -> D, F -> GH, I -> JGH*.
- *What is the closure of ABC, ABC^+?*

1. Basis $ABC^+ = ABC$
2. $ABC^+ = ABC^+$ union *D*, since ABC is a subset of ABC^+ and *ABC -> D*
3. $ABC^+ = ABC^+$ union *F*, since ABCD is a subset of ABC^+ and *ABCD -> F*

Example – Closure Test

- Assume *ABCD -> F, ABC -> D, F -> GH, I -> JGH*.
- *What is the closure of ABC, ABC^+?*

1. Basis $ABC^+ = ABC$
2. $ABC^+ = ABC^+$ union *D*, since ABC is a subset of ABC^+ and *ABC -> D*
3. $ABC^+ = ABC^+$ union *F*, since ABCD is a subset of ABC^+ and *ABCD -> F*
4. $ABC^+ = ABC^+$ plus *GH*, since *F* is a subset of ABC^+ and *F -> GH*

Therefore $ABC^+ = ABCDFGH$.

Hence, for instance, ABC -> H is true but ABC -> I is not

Boyce-Codd Normal Form

We say a relation *R* is in *BCNF* if whenever *X ->Y* is a nontrivial FD that holds in *R*, *X* is a *superkey*.

- *nontrivial* means *Y* is not contained in *X*.
- Remember, a *superkey* is any superset of a key

Decomposition into BCNF

Given: relation *R* with FD's *F*.

Look among the given FD's for a BCNF violation: $X \rightarrow Y$.

- Compute X^+

Decompose *R* Using *X* -> *Y*

Replace *R* by relations with schemas:

1. $R_1 = X^+$.
2. $R_2 = R - (X^+ - X) = R - X^+ + X$.

Project given FD's *F* onto the two new relations.

Third Normal Form -- Motivation

There is one structure of FD's that causes trouble when we decompose tables into subtables.

AB ->*C* and *C* ->*B*.

- Example: *A* = street address, *B* = city, *C* = zip code.

There are two keys, {*A*,*B*} and {*A*,*C*}.

Is there a BCNF violation? What is the fix?

Third Normal Form -- Motivation

There is one structure of FD's that causes trouble when we decompose tables into subtables.

AB ->C and *C ->B*.

- Example: *A* = street address, *B* = city, *C* = zip code.

There are two keys, {*A*,*B*} and {*A*,*C*}.

C ->B is a BCNF violation, so we must decompose into *AC*, *BC*.

We Cannot Enforce FD’s

The problem is that if we use *AC* and *BC* as our database schema, we cannot enforce the FD *AB ->C* by checking FD’s in these decomposed relations.

Example with *A* = street, *B* = city, and *C* = zip on the next slide.

An Unenforceable FD

street	zip
545 Tech Sq.	02138
545 Tech Sq.	02139

city	zip
Cambridge	02138
Cambridge	02139

Join tuples with equal zip codes.

street	city	zip
545 Tech Sq.	Cambridge	02138
545 Tech Sq.	Cambridge	02139

Although no FD's were violated in the decomposed relations, FD street city -> zip is violated by the database as a whole.

3NF Let Us Avoid This Problem

3rd Normal Form (3NF) modifies the BCNF condition so we do not have to decompose in this problem situation.

An attribute is *prime* if it is a member of any key.

X ->*A* violates 3NF if and only if *X* is not a superkey, and also *A* is not prime.

Example: 3NF

In our problem situation with FD's *AB ->C* and *C ->B*, we have keys *AB* and *AC*.

Thus *A*, *B*, and *C* are each prime.

Although *C ->B* violates BCNF, it does not violate 3NF.

What 3NF and BCNF Give You

There are two important properties of a decomposition:

1. *Lossless Join* : it should be possible to project the original relations onto the decomposed schema, and then reconstruct the original.
2. *Dependency Preservation* : it should be possible to check in the projected relations whether all the given FD's are satisfied.

3NF and BCNF -- Continued

We can get (1) with a BCNF decomposition.

We can get both (1) and (2) with a 3NF decomposition.

But we can’t always get (1) and (2) with a BCNF decomposition.

- street-city-zip is an example.

3NF Synthesis Algorithm

We can always construct a decomposition into 3NF relations with a lossless join and dependency preservation.

Need *minimal basis* for the FD's:

1. Right sides are single attributes.
2. No FD can be removed.
3. No attribute can be removed from a left side.

Constructing a Minimal Basis

1. Split right sides.
2. Repeatedly try to remove an FD and see if the remaining FD's are equivalent to the original.
3. Repeatedly try to remove an attribute from a left side and see if the resulting FD's are equivalent to the original.

3NF Synthesis – (2)

One relation for each FD in the minimal basis.

- Schema is the union of the left and right sides.

If no key is contained in an FD, then add one relation whose schema is some key.

Example: 3NF Synthesis

Relation R = ABCD.

FD's *A->BC* is equivalent to:

FD's *A->B* and *A->C*.

Decomposition: AB and AC from the FD's, plus AD for a key.

Why It Works

Preserves dependencies: each FD from a minimal basis is contained in a relation, thus preserved.

Lossless Join: yes

3NF: yes.

Actions

Review slides!

Read Chapters 3.1. – 3.5 (Design Theory for Relational Databases).

Information Integration

Information integration is the process of taking several databases and making the data in these sources work together as if they were a single database.

The integrated database may be

- Physical (”data warehouse”)
- Virtual (”mediator”) that may be queried even though it does not exist physically

Information-integration systems require special kinds of query-optimization techniques for their efficient operation.

Why Information Integration?

If we could put data always in a single database, there would be no need for information integration.

However, in the real world, matters are rather different..

- Databases are created intependently, even if they later need to work together.
- The use of databases evolves, so we cannot design a database to support every possible future use.

Example Applications

1. Enterprise Information Integration: making separate DB’s, all owned by one company, work together.
2. Scientific DB’s, e.g., genome DB’s.
3. Catalog integration: combining product information from all your suppliers.

Challenges

1. *Legacy databases* : DB’s get used for many applications.
   - You can’t change its structure for the sake of one application, because it will cause others to break.

2. *Incompatibilities* (heterogenity problem): Two, supposedly similar databases, will mismatch in many ways.

Examples: Incompatibilities

Lexical : `addr` in one DB is `address` in another.

Value mismatches : is a “BL” car the same color in each DB (blue versus black)? Is 20 degrees Fahrenheit or Centigrade?

Semantic : are “employees” in each database the same? What about consultants? Retirees? Contractors?

Query-Language heterogenity : Relational database (SQL) verus XML (Xquery)

Data Type differences : Serial numbers might be represented as *string* in one source and *integer* in another source.

Examples: Schema Heterogeneity

One dealer might store cars in a single relation that look like:

- `Cars(serialNo, model, color, autoTrans, navi, ...)`

Another dealer might use a schema in which options are seperated out into a second relation, such as:

- `Autos (serial, model, color)`
- `Options (serial, option)`

What Do You Do About It?

Grubby, handwritten translation at each interface.

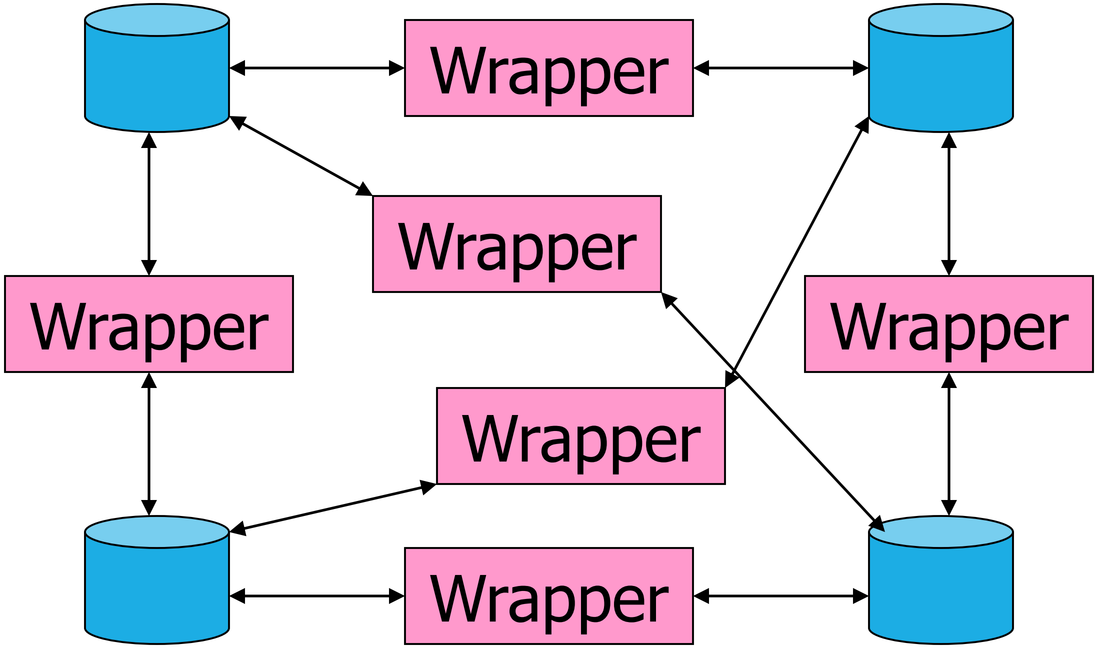
- Some research on automatic inference of relationships.

Wrapper (aka “adapter”) translates incoming queries and outgoing answers.

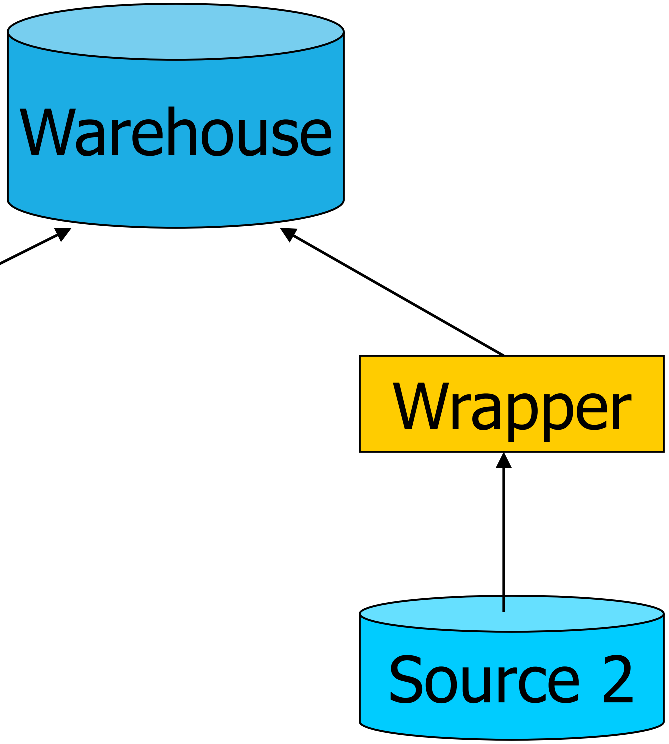
Integration Architectures

1. *Federation* : everybody talks directly to everyone else.
2. *Warehouse* : Sources are translated from their local schema to a global schema and copied to a central DB.
3. *Mediator* : *Virtual warehouse* --- turns a user query into a sequence of source queries.

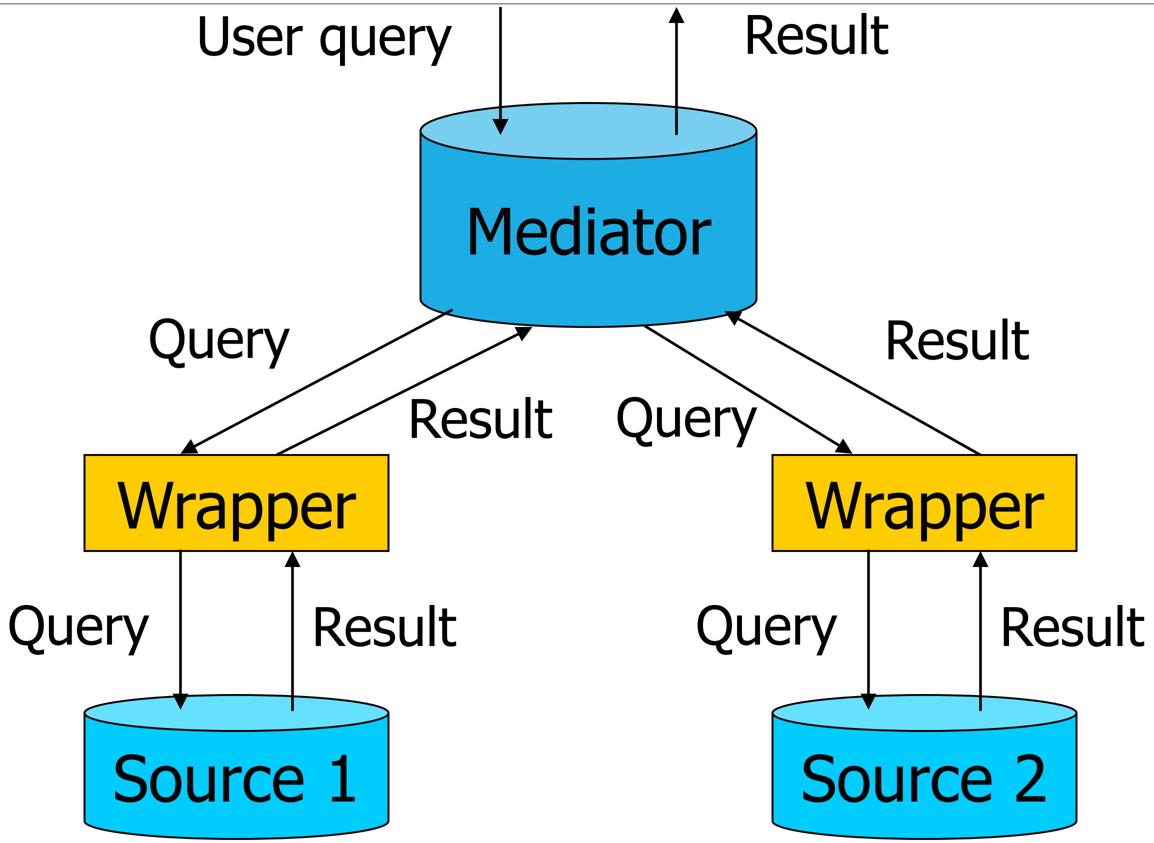
Federations



Warehouse Diagram



A Mediator


User query
Result
Mediator
Query
Result
Query
Result
Wrapper
Wrapper
Query
Result
Query
Result
Source 1
Source 2

Example: Mediatior

Suppose mediator integrates the same two automobile sources into a view that is a single relation with schema:

- `AutosMed (serialNo, model, color, autoTrans, dealer)`

Assume the user asks the mediator about red cars, with the query:

```
SELECT serialNo, model
FROM AutosMed
WHERE color = ’red’;
```

Schema Heterogeneity

One dealer might store cars in a single relation that look like:

- `Cars(serialNo, model, color, autoTrans, navi, ...)`

Another dealer might use a schema in which options are seperated out into a second relation, such as:

- `Autos (serial, model, color)`
- `Options (serial, option)`

Example: Mediatior

The wrapper for Dealer 1 translates the query into the terms of the dealer's schema:

```
SELECT SerialNo, model
FROM Cars
WHERE color = 'red'
```

At the same time, the wrapper for Dealer 2 translates the same query into the schema of that dealer:

```
SELECT serial, model

FROM Autos

WHERE color = 'red';
```

The mediator takes union of these sets and returns the result to the user.

Mediation Approach

Mediator processes queries into steps executed at sources.

Example: Catalog Integration

Suppose Dell wants to buy a bus and a disk that share the same protocol.

Global schema: `Buses(manf,model,protocol)`
`Disks(manf,model,protocol)`

Local schemas: each bus or disk manufacturer has a (model,protocol) relation --- manf is implied.

Example: Global-as-View

Mediator might start by querying each bus manufacturer for model-protocol pairs.

- The wrapper would turn them into triples by adding the manf component.

Then, for each protocol returned, mediator queries disk manufacturers for disks with that protocol.

- Again, wrapper adds manf component.

Actions

Read Chapter *Information Integration* (21.1-2)