【VLDB2010勉強会】

Session 16: Query Processing I

担当：渡辺陽介（東京工業大学）

[1件目] Slicing Long-Running Queries

- N. Bruno, V. Narasayya, R. Ramamurthy (Microsoft)
- 概要
  - DBMS上での高コストの問合せ実行が制限される場合がある
    - リソースの独占回避のためのアドミッション制御
  - 耐故障性，負荷分散の観点から，小さい処理単位を部分的に実行できるようにしたい

⇒単体ではコストが高すぎる問合せを，一定閾値以下のコストに収まる部分問合せに分解する問題 (Query slicing problem)

- 部分問合せ間で中間結果を受け渡すための演算(Spool)を定義
- 最適な部分問合せへの分割手法の提案と評価

問合せの分割例

元の問合せ

```
SELECT  R.a, S.b
FROM  R  JOIN  S  ON R.x=S.y
WHERE   R.c < 10  AND  S.d > 20
```

Rの選択演算を単独実行

```
INSERT   INTO   TR
SELECT  R.a, R.x
FROM  R
WHERE   R.c < 10
```

```
SELECT  TR.a, S.b
FROM  TR  JOIN  S  ON TR.x=S.y
WHERE   S.d > 20
```

Sを属性dの値で水平分割

```
SELECT  R.a, S.b
FROM  R  JOIN  S  ON R.x=S.y
WHERE   R.c < 10  AND  S.d > 20
            AND   S.d  < 100
```

```
SELECT  R.a, S.b
FROM  R  JOIN  S  ON R.x=S.y
WHERE   R.c < 10  AND   S.d  >= 100
```

全体では元の問合せよりもコストは高くなってしまうが，個々の部分問合せのコストは安い

Spool演算

- Spool演算
  - 入力された中間結果を一時テーブルに保存する演算
- Input-partitioned Spool演算 (iSpool)
  - 値で分割された中間結果を入力として受け取るSpool演算
- Output-partitioned Spool演算 (oSpool)
  - 入力された中間結果を値で分割した一時テーブルに保存するSpool演算
- Input/Output partitioned Spool演算 (ioSpool)
  - 分割された入力を受け取り，別の分割条件で一時テーブルに保存するSpool

Figure 2(a)

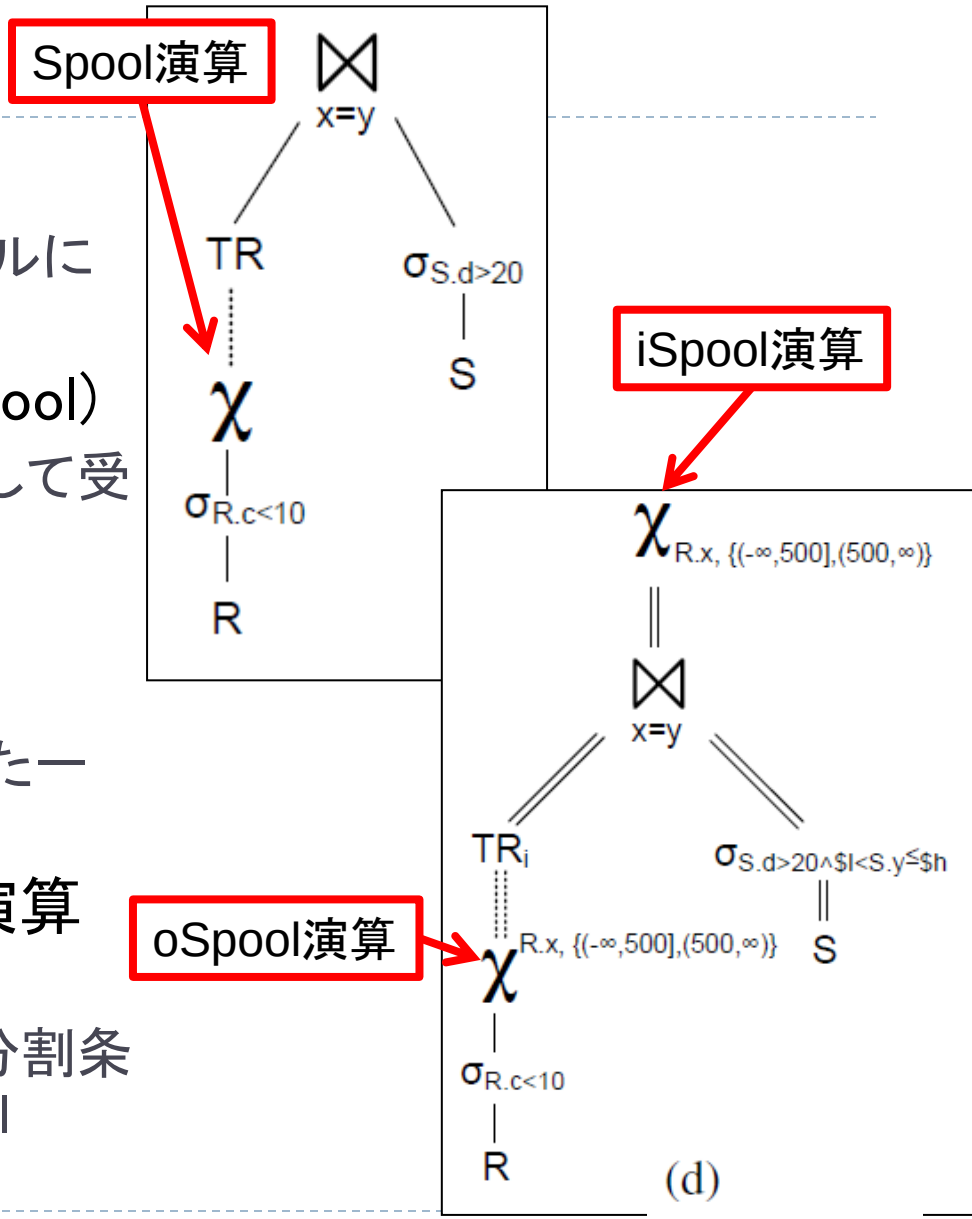
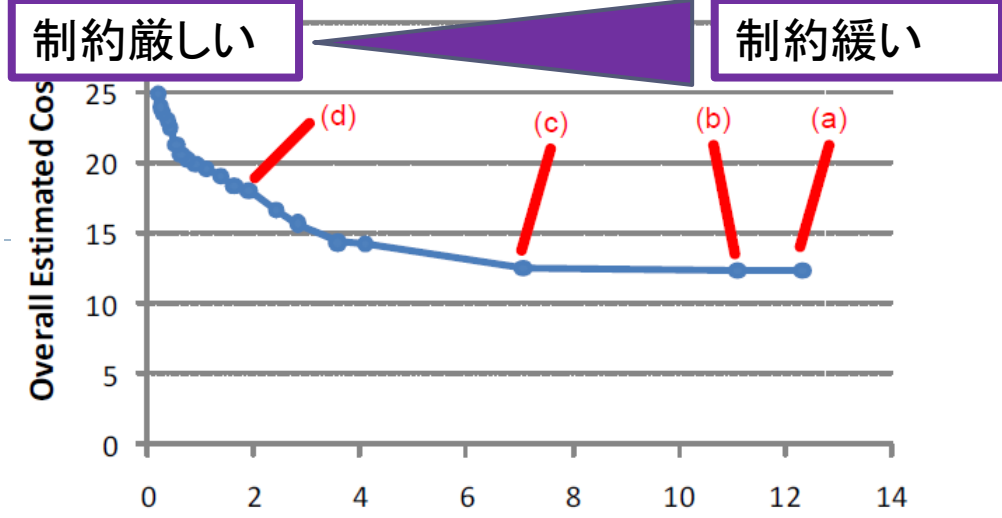


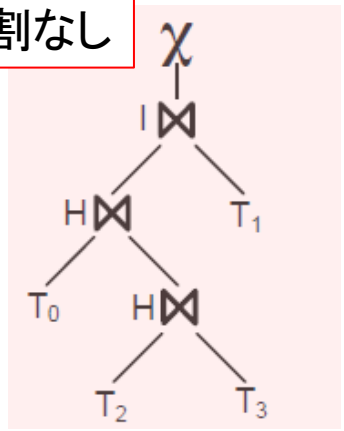
Figure 2(d)

閾値Δ を変えた実験

Figure 10

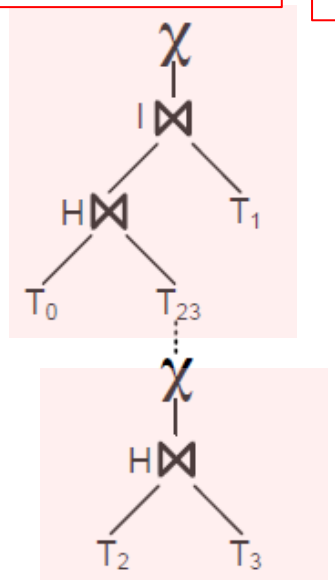


分割なし



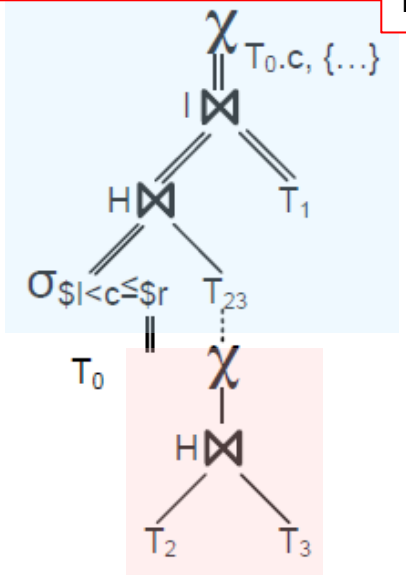
(a) Δ=12.4, C=12.5

Spoolで分割



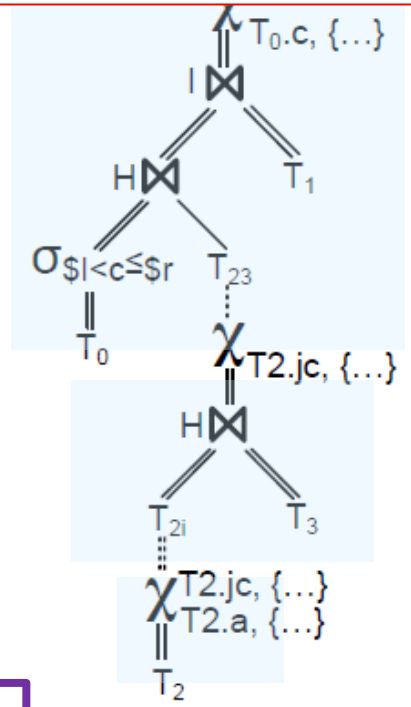
(b) Δ=11, C=12.6

Spool,iSpoolで分割



(c) Δ=7, C=13.1

ioSpool,iSpool,iSpoolで分割



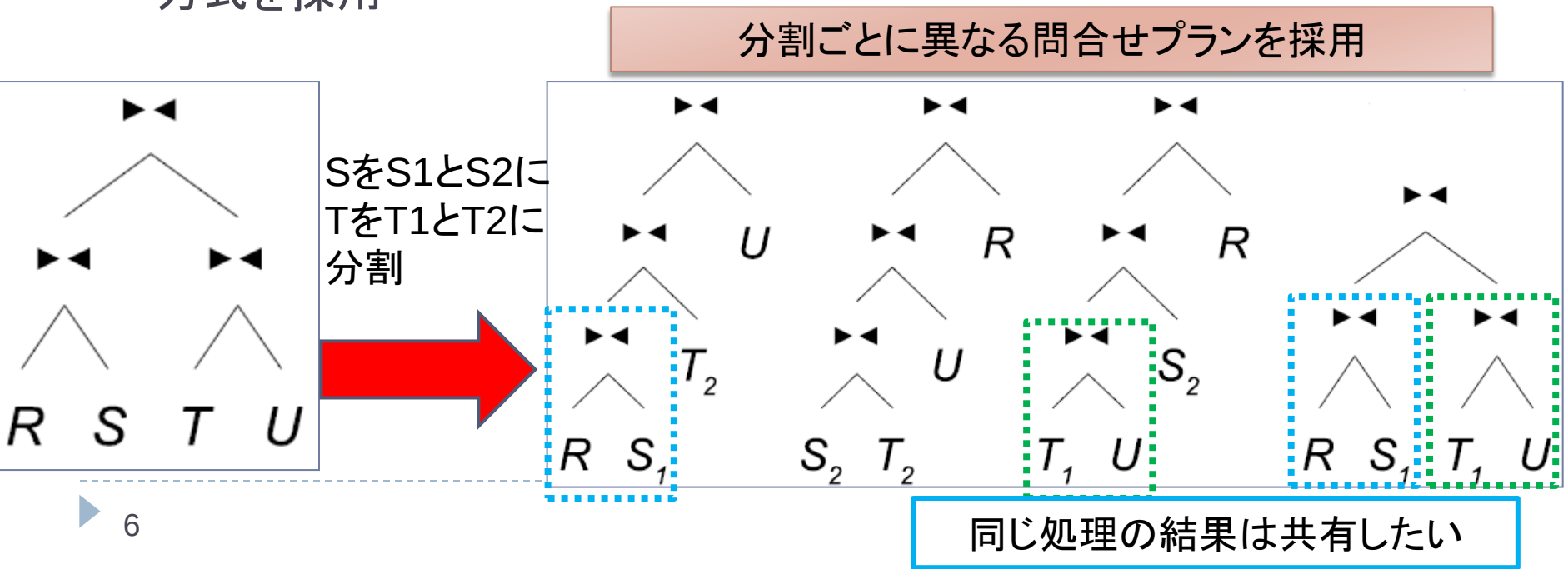
(d) Δ=1.9, C=18.1

制約緩い

制約厳しい

[2件目] Sharing-Aware Horizontal Partitioning for Exploiting Correlations During Query Processing

- K. Tzouma (Aalborg univ.), A. Deshpande (Univ. Maryland), C. S. Jensen (Aarhus Univ.)
- データの水平分割による多段joinの最適化手法
  - データ同士の相関に基づいて水平分割を決定
  - 分割ごとのデータの統計的偏りの違いを考慮し，別々の問合せプランを用いる
  - 複数の問合せプランの間で冗長な中間結果が大量に出ないような実行方式を採用

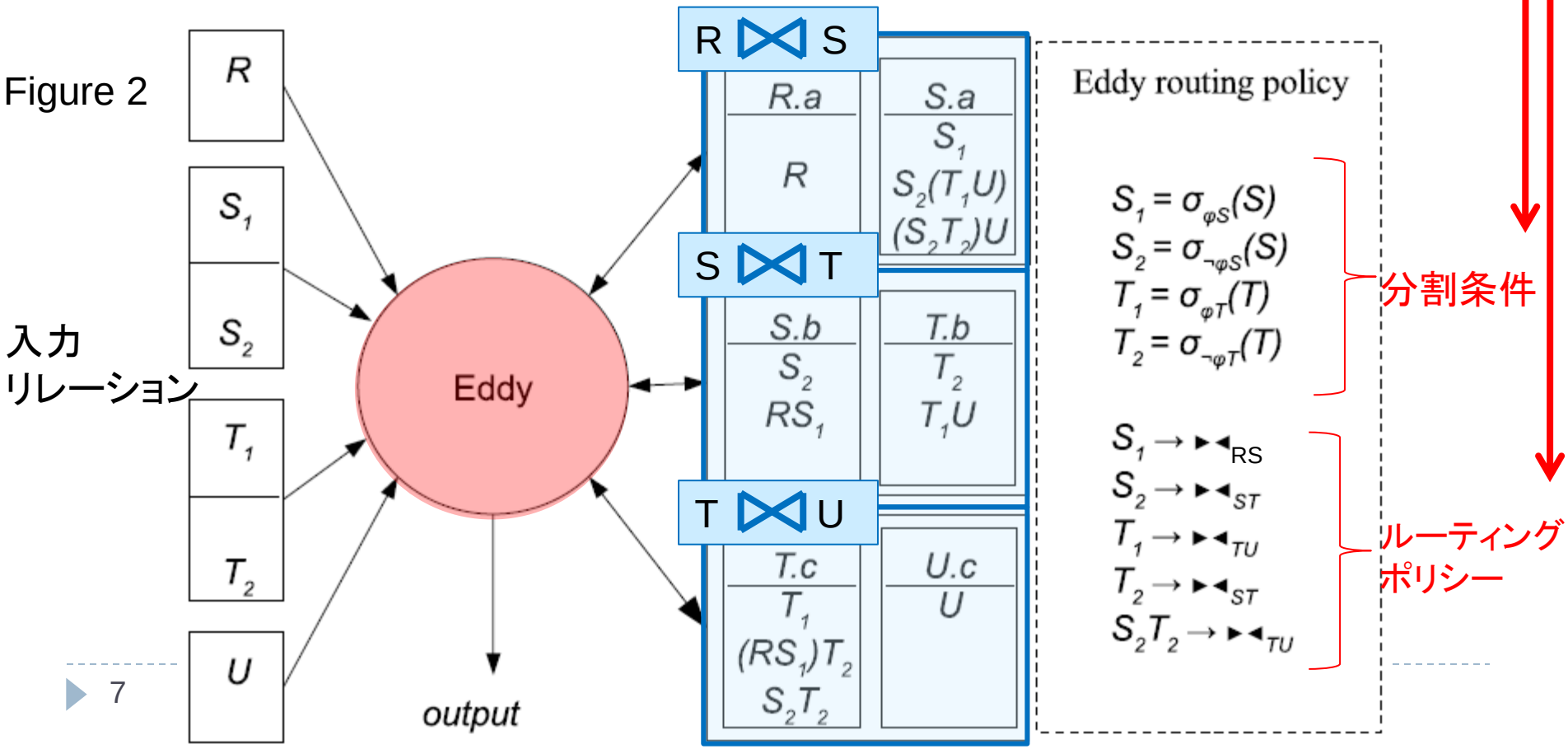


複数問合せプランの処理機構

- Eddy [SIGMOD2000] の採用
  - タプルごとに演算の通過順を決定できる演算
    - この研究では分割ごとにルーティングポリシーを決定
  - 複数の問合せプランをルーティングポリシーとして記述する

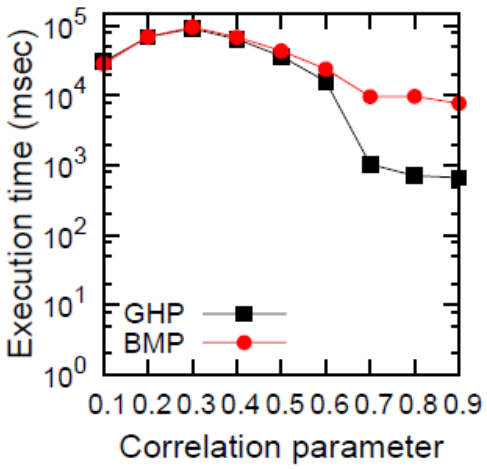
最適化問題:
これらをどのように
決定するか

Figure 2

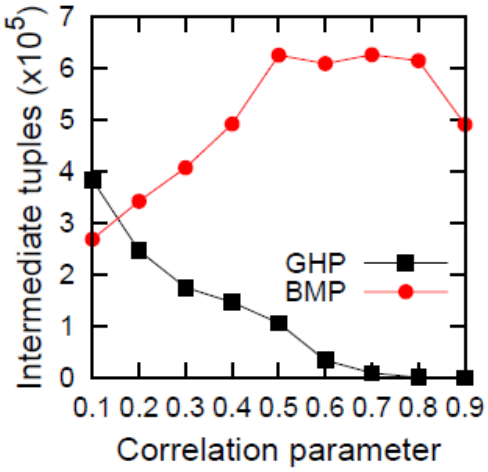


最適化手法

- コスト見積り [Section 3]
  - 相関のあるデータからの選択率の見積り
    - トレーニングデータから各属性値の確率分布を表すMarkov networkを構築し，joint probabilityを計算
- Greedy Horizontal Partition Algorithm (GHP) [Appendix C]
  - 最適なデータ分割と，その分割において最適な問合せプランの両方を同時に探索する



(a) Total execution time.



(b) Intermediate tuples generated.

Figure 5

提案手法
GHP

比較対象
Best Monolithic plan (BMP)

[3件目] Advanced Processing for Ontological Queries

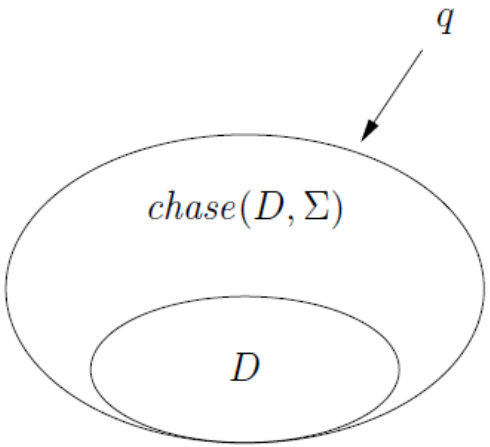
- A. Cali, G. Gottlob (Unix. Oxford), A. Pieris (Brunel Univ.)
- Ontological Database Management System
  - オントロジー・記述論理とデータベース技術の融合
    - Semantic Web分野
    - Quonto [4], FaCT [23]などの研究ベースのシステム
- 提案内容
  - DL-Liteという記述論理を拡張した言語の提案
  - Sticky tuple-generating dependency (TGD)を定義
    - TGD
      - データベース内に明示的に存在しなくても，ルールを適用するとタプルが導出できる性質
      - 「Johnという人間がいるなら，Johnの父にあたる人間もいるだろう」
    - 無限にタプルが導出されないように変数の記述に制限を設定
    - 複数リレーションのJoinを使うTGDを記述可能にした

Tuple-Generationg Dependency (TGD)

- D: データベース
- Σ : 記述論理で書かれたルール集合 (これがTGDsか?)
- chase(D, Σ)
  - データベースDにルールΣ を適用すると，元のデータベースには明示的に存在しなかったタプルが生まれる

D

$$dept(\text{Dept\_Id}, \text{Mgr\_Id}),$$
$$emp(\text{Emp\_Id}, \text{Dept\_Id}, \text{Area}, \text{Project\_Id}),$$
$$runs(\text{Dept\_Id}, \text{Project\_Id}),$$
$$in\_area(\text{Project\_Id}, \text{Area}),$$
$$project\_mgr(\text{Emp\_Id}, \text{Project\_Id}),$$
$$external(\text{Ext\_Id}, \text{Area}, \text{Project\_Id}).$$



Σ

$$dept(V, W) \rightarrow \exists X \exists Y \exists Z \; emp(W, X, Y, Z).$$

部門Vには，その部門Vの部長である社員Wが存在する

$$runs(W, X), in\_area(X, Y) \rightarrow \exists Z \; external(Z, Y, X).$$

Y分野のプロジェクトXを遂行する部門Wには，Y分野担当の外部監査官Zがつく