AMPL：数理計画のためのモデリング言語



第10章

データベースアクセス

table 宣言で *AMPL* の集合・パラメータ・変数を関係データベースの表に対応づける

table 宣言

read table / write table

ODBC・Access・Excel・テキスト

本章の内容

AGENDA

1. **データ対応の一般原則**
   AMPLの実体と関係表の列の対応
2. **テーブル操作文の例**
   table宣言・read table・write table
3. **関係表からのデータの読み込み**
   キー列・データ列・集合の読込
4. **関係表へのデータの書き出し**
   data-spec / key-spec からの推論
5. **同じ表の読み書き**
   IN・OUT・INOUTの使い分け
6. **表と列の添字づけられた集まり**
   テーブル配列と2次元表の入出力
7. **標準および組込みのテーブルハンドラ**
   ODBC・テキスト・バイナリ・SQL

AMPLの実体と関係表の対応

10.1

diet.mod

```
set FOOD;

param cost {FOOD} > 0;
param f_min {FOOD} >= 0;
param f_max {j in FOOD}
   >= f_min[j];
```

- 集合 FOOD の要素 → 表の「キー列」
- パラメータ cost / f_min / f_max → 各1本の「データ列」
- 1行 = 集合の1要素（1食品）のすべてのデータ

FOOD	cost	f_min	f_max
BEEF	3.19	2	10
CHK	2.59	2	10
FISH	2.29	2	10
HAM	2.89	2	10
…	…	…	…

キー列 *FOOD* の項目は一意 ＝ 各キー値がちょうど*1*行を特定する

行 = レコード　列 = フィールド

データベース用語では、表の各行はレコード、レコード内の各項目はフィールドと呼ばれる。

レコード単位のデータ入力

10.1

- キー列の項目は集合の要素と同様に一意でなければならない
- データベースソフトでは、表の行は「レコード」、列は「フィールド」とみなされる
- データ入力フォームには列ごとに1つの入力欄が用意される
- 配合問題のフォーム（Microsoft Access）では、表の各行に対応するレコードを、フォーム下部の操作部で1件ずつ入力・閲覧できる

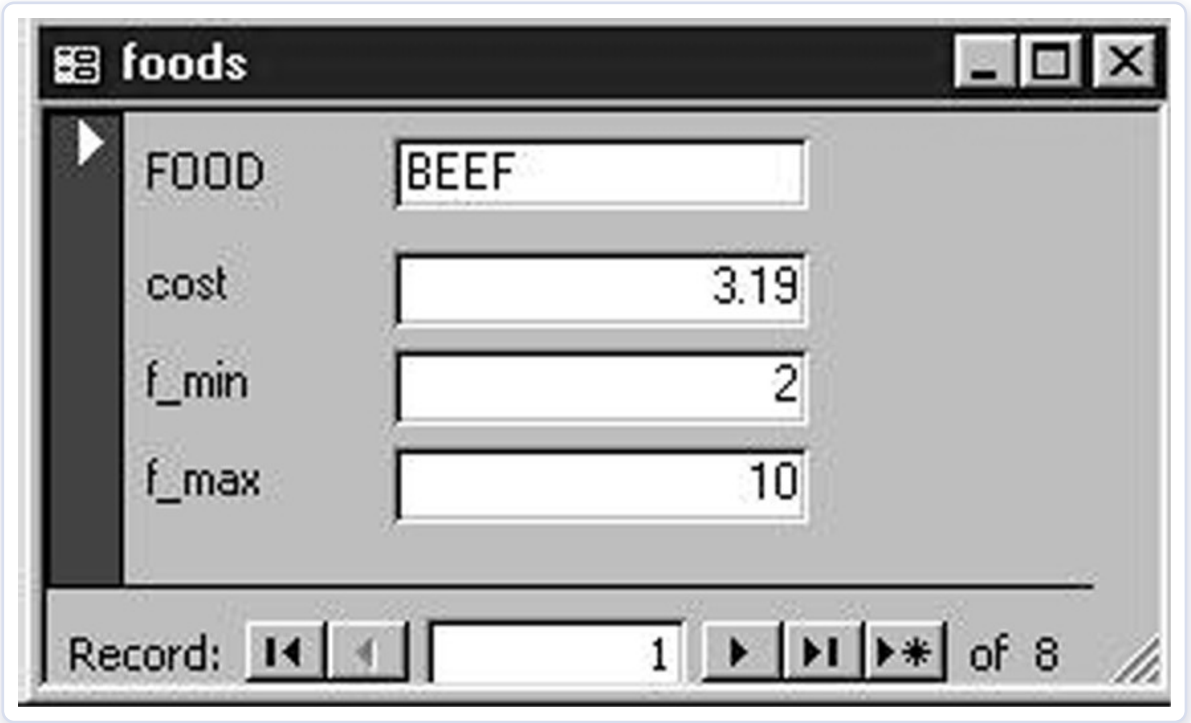


図10-1　Accessのデータ入力フォーム

多次元の実体 → 複数のキー列

10.1

steelT.mod

```
set PROD;       # products
param T > 0;    # number of weeks

param market {PROD,1..T} >= 0;
param revenue {PROD,1..T} >= 0;
var Make {PROD,1..T} >= 0;
```

- PROD と 1..T の直積で添字づけられた実体 → キー列が2つ (PROD, TIME)
- 各次元につき1つのキー列を割り当てる
- 見出し TIME は式 1..T を表すために考案した名前

PROD	TIME	market	revenue	Make	Sell
bands	1	6000	25	5990	6000
bands	2	6000	26	6000	6000
coils	1	4000	30	1407	307
coils	2	2500	35	1400	2500

2つのキー列の順序対は一意

集合 {PROD,1..T} で対が一意であるのと同様、表でも ("bands",1) や ("coils",3) のような組はちょうど1行に対応する。

データで決まる集合との対応

10.1

transp3.mod

```
set LINKS within {ORIG,DEST};
param cost {LINKS} >= 0;
var Trans {LINKS} >= 0;
```

- LINKS は1次元集合から構成されるのではなく、データとして読まれる対の集合
- 対応する関係表もキー列を2つもつ（ORIG, DEST）
- 表には ORIG・DEST の全組合せではなく、実際に集合 LINKS に属する組だけが行として現れる

ORIG	DEST	cost	Trans
GARY	DET	14	0
GARY	LAF	8	600
CLEV	DET	9	1200
PITT	STL	28	900
PITT	WIN	13	0

網羅的な直積表ではなく「実在する組だけ」を並べる疎な表

table宣言の構文構造

10.2

```
table table-name IN string-list :  key-spec, data-spec, data-spec, ... ;
```

テーブル名

AMPL内でこの表を指す名前（例：Foods）

IN / OUT

既定の読み書き状態。省略時は INOUT

string-list

外部ファイル・表の場所（ハンドラ固有）

key-spec

キー列 [...] と集合との対応（<-, ->, <->）

data-spec

データ列とパラメータ・変数・式との対応

コロンより前＝表の所在（テーブル名・*IN/OUT・外部ファイル*）　／　*コロンより後＝列とAMPLの実体との対応*

同じ対応、異なる外部形式

10.2

Access

```
table Foods IN "ODBC" "diet.mdb":
   FOOD <- [FOOD], cost, f_min,
f_max;
```

diet.mdb

Excel

```
table Foods IN "ODBC" "diet.xls":
   FOOD <- [FOOD], cost, f_min,
f_max;
```

diet.xls

テキスト

```
table Foods IN:
   FOOD <- [FOOD], cost, f_min,
f_max;
```

Foods.tab

- コロン前の情報だけが変わる：ODBCハンドラ経由か、既定のテキストハンドラかを指定
- 3つとも AMPL 側のテーブル名は同じ Foods → read table Foods は共通
- 詳細を省略すると既定でテキストファイル Foods.tab が使われる

read table でAMPLに読み込む

amplセッション

```
ampl: model diet.mod;
ampl: table Foods IN "ODBC" "diet.mdb":
ampl?    FOOD <- [FOOD], cost, f_min, f_max;
ampl: read table Foods;
ampl: display cost, f_min, f_max;
:      cost f_min f_max    :=
BEEF   3.19    2    10
CHK    2.59    2    10
FISH   2.29    2    10
HAM    2.89    2    10
  ...
;
```

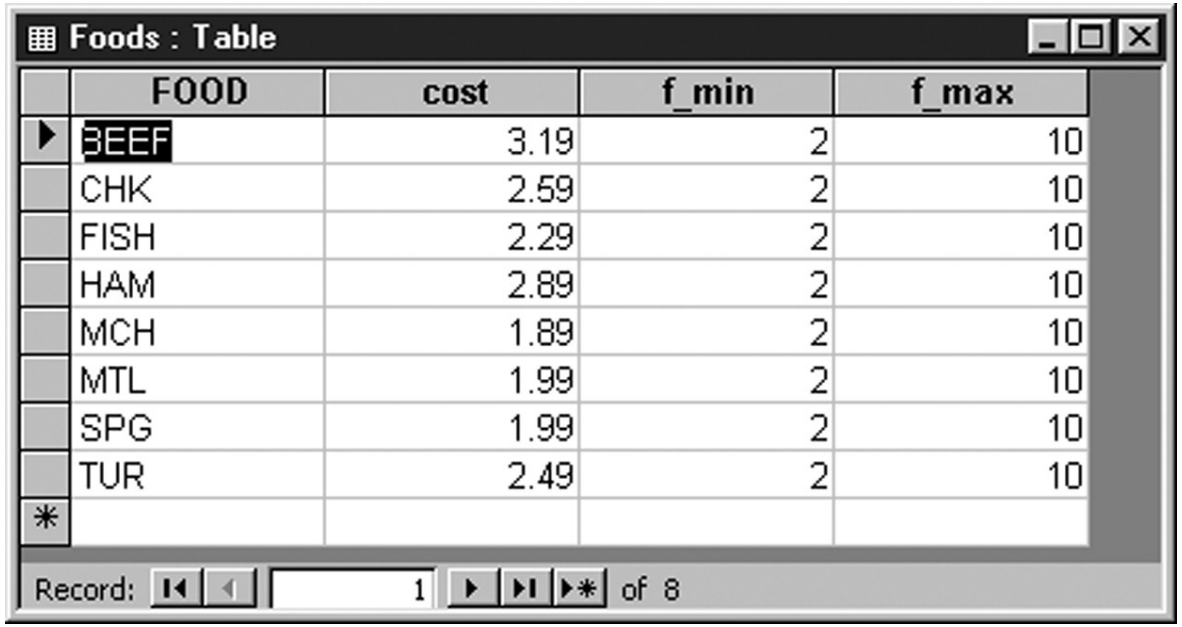


図10-2　Accessの関係表

入力と出力を1つの宣言に


```
table Foods "ODBC" "diet.mdb":
   [FOOD] IN, cost IN,
   f_min IN, f_max IN, Buy OUT,
   Buy.rc ~ BuyRC OUT,
   {j in FOOD} Buy[j]/f_max[j]
      ~ BuyFrac OUT;
```

Foods : Table

FOOD	cost	f_min	f_max	Buy	BuyRC	BuyFrac
BEEF	3.19	2	10	5.360613811	3.05311E-16	0.536061381
CHK	2.59	2	10	2	1.18884058	0.2
FISH	2.29	2	10	2	1.144407502	0.2
HAM	2.89	2	10	10	-0.30265132	1
MCH	1.89	2	10	10	-0.5511509	1
MTL	1.99	2	10	10	-1.32890026	1
SPG	1.99	2	10	9.306052856	-2.0123E-15	0.930605286
TUR	2.49	2	10	2	2.731619778	0.2

Record: 1 of 8

図10-4　出力列が加わったAccessの関係表

- IN / OUT を表全体ではなく列ごとに指定できる
- ~ 演算子で AMPL の式（Buy.rc など）をデータベースの正しい列名に結びつける
- read table → IN列を読み、write table → OUT列に書き出す

データ読み込みの2つの基本形

10.3

パラメータのみ読む

```
table Foods IN:
   [FOOD], cost, f_min, f_max;
```

- 集合 FOOD はモデルまたは他の表であらかじめ定義されている
- キー列の項目を各パラメータの添字として値を割り当てるだけ

集合とパラメータを同時に読む

```
table Foods IN:
   FOOD <- [FOOD], cost, f_min, f_max;
```

- 矢印 <- はキー列 → 集合への読み込み方向を示す
- key-spec を短縮形 [FOOD] IN と書くこともできる（集合名=キー列名の場合）

列名の対応づけとダミー添字による調整

10.3

列名が違う

*AMPL*側と表側で名前が食い違う場合

```
f_min ~ lowerlim, f_max ~ upperlim
```

番号の基準が違う

表は0始まり、モデルは1始まりなどのずれを補正

```
[p ~ PROD, t ~ TIME], market[p,t+1] ~ market
```

添字の順序が違う

同じ集合でも添字の並びが異なる場合

```
market, revenue[t,p] ~ revenue
```

1つの値が複数列に分割

系列ごとに別列で保持されている場合

```
revenue["bands",t] ~ revbands, revenue["coils",t] ~ revcoils
```

データ指定(data-spec)から行を推論

10.4

```
table Foods OUT: [FoodName],
   f_min, Buy, f_max;
```

- key-spec が単なる [キー列名] のリストのとき、display コマンドと同様に動く
- data-spec が示す（あるいは含意する）添字集合の和で、書き出す行が決まる
- 挙げる項目はすべて同じ次元でなければならない

FoodName	f_min	Buy	f_max
BEEF	2	5.36061	10
CHK	2	2	10
HAM	2	10	10
SPG	2	9.30605	10

write table Foods; → FOOD の各要素につき1行が生成される

キー指定(key-spec)から行を推論

10.4

```
table Foods OUT:
   FOOD -> [FOOD],
   f_min, Buy, f_max;
```

- 矢印 -> は AMPL の集合 → キー列へ書き出す方向
- 明示した集合の各要素につき1行を書き出す（data-specの添字集合からの推論ではない）
- 同じ集合名がキー列名なら [FOOD] OUT と略記できる

! エラーになる例

```
table Nutrs OUT: [Nutrient],
   Diet.lslack, Diet.ldual, ...;
# Column "Diet.lslack" contains '.'
```

✓ ~ で列名を与えて解決

```
table Nutrs OUT: [Nutrient],
   Diet.lslack ~ lb_slack,
   Diet.ldual ~ lb_dual,
   Diet.uslack ~ ub_slack,
   Diet.udual ~ ub_dual;
```

↔ 同じ表を読み書きする

10.5

- 方法A：読み込み用と書き出し用、2つの table 宣言を使う
- 方法B：1つの table 宣言に読み書きの状態をまとめる

指定	read table	write table
<- / IN	読む	—
-> / OUT	—	書く
<-> / INOUT	読む	書く

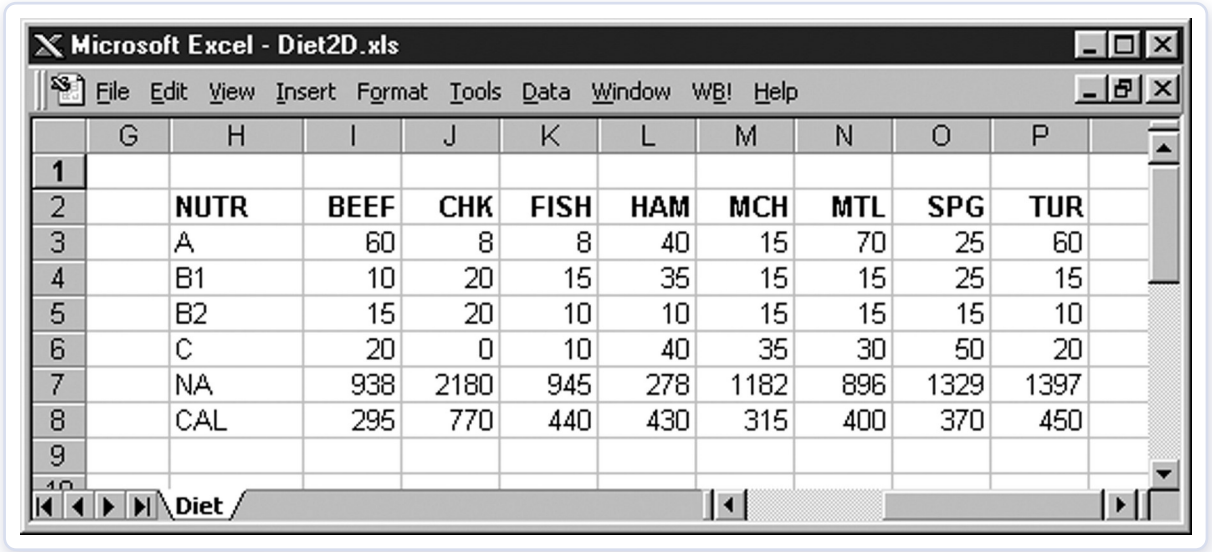
単一の宣言

```
table Foods "ODBC" "diet1.mdb" "Diet":
   FOOD <- [FoodName],
   cost IN, f_min IN,
   Buy OUT, f_max IN;
```

- read table Foods → IN・INOUT状態の列だけ読む（FoodName, cost, f_min, f_max）
- write table Foods → OUT・INOUT状態の列だけ書く（Buy）
- 既存の表・ファイル全体を上書きするのは、全列がOUTのときだけ

テーブル・列の添字づけられた集まり

- table 宣言も {indexing-expr} で添字づけできる（例：食品ごとに別々の表へ出力）
- data-spec も {j in FOOD} <…> の形で列の集まりを生成できる
- 二次元の表計算データ（行×列）を読み書きするのに特に便利



	G	H	I	J	K	L	M	N	O	P
1										
2		NUTR	BEEF	CHK	FISH	HAM	MCH	MTL	SPG	TUR
3		A	60	8	8	40	15	70	25	60
4		B1	10	20	15	35	15	15	25	15
5		B2	15	20	10	10	15	15	15	10
6		C	20	0	10	40	35	30	50	20
7		NA	938	2180	945	278	1182	896	1329	1397
8		CAL	295	770	440	430	315	400	370	450
9										

図10-7　Excelにおける二次元のAMPLテーブル

2次元表の読込

```
table dietAmts IN "ODBC" "Diet2D.xls":
   [i ~ NUTR],
   {j in FOOD} <amt[i,j] ~ (j)>;
```

標準および組込みのテーブルハンドラ

10.7

- **標準 ODBC ハンドラ** — Access・Excel など ODBC ドライバを持つあらゆるアプリと接続（Windows）
- **組込みテキスト／バイナリ** — .tab（ASCII）／.bit（バイナリ）— デバッグ・実演用の最小実装
- **SQL クエリの埋め込み** — "SQL=SELECT …" で一時的な関係表を作ってから読み込める

SQL埋め込み例

```
table cheapFoods IN "ODBC" "diet.mdb"
   "SQL=SELECT * FROM Foods WHERE cost <= 2.49":
   FOOD <- [FOOD], cost, f_min, f_max;
```

✓ まとめ：第10章 データベースアクセス

1. table 宣言が、AMPLの集合・パラメータ・変数・式と、関係表の列との対応を定義する
2. 実際の入出力は read table / write table コマンドで実行される
3. キー列は <- ・ -> ・ <-> で、データ列は IN・OUT・INOUT で読み書きの方向を制御する
4. ~ 演算子とダミー添字により、列名の違い・番号のずれ・添字順序の違いに対応できる
5. 標準ODBCハンドラに加え、テキスト・バイナリの組込みハンドラやSQLクエリも利用できる